Overload control of telephone exchanges

Waal, Petrus Rosalia de

Publication date:
1990

Document Version
Publisher's PDF, also known as Version of record

Peter de Waal

Overload Control of Telephone Exchanges

Overload Control of Telephone Exchanges

Overload Control of Telephone Exchanges

PROEFSCHRIFT
ter verkrijging van de graad van doctor
aan de Katholieke Universiteit Brabant,
op gezag van de rector magnificus,
prof. dr. R.A. de Moor,
in het openbaar te verdedigen
ten overstaan van
een door het college van dekanen aangewezen commissie
in de aula van de Universiteit op
donderdag 5 juli 1990 te 16.15 uur
door

PETRUS ROSALIA DE WAAL

geboren te Kloosterzande.

Promotor: Prof. dr. ir. O.J. Boxma
Assistent-promotor: Dr. ir. J.H. van Schuppen

Preface

The research work that is presented in this thesis is the result of the project 'Overload Control of Communication Systems' that was supported by the Technology Foundation (STW) and was carried out at the Centre for Mathematics and Computer Science (CWI) in Amsterdam. I would like to thank both organisations for the generous support that I have received.

A large number of people have contributed to the successful completion of this thesis of whom I would like to mention two. It is a great pleasure to thank Jan H. van Schuppen for being the supervisor in this research project. His continuous interest and stimulating enthusiasm have been a constant support during these years.

I am grateful to both Jan H. van Schuppen and Onno Boxma for the valuable comments on the many draft versions of this thesis.

Contents

1

Introduction

In a modern technological society new services like data communication and video networks are becoming increasingly important. The main part of today's communication, however, is still carried by the telephone network. Usually we do not realise how much we rely on this means of communication, until the network is temporarily not available. Such an unavailability can be caused by technical failures or by a demand for telephone services that exceeds the capacity of the network and the exchanges. The latter situation is called overload and the problems that it may cause are of great concern to telephone companies.

We can distinguish between incidental and structural occurrences of this phenomenon. Structural overload periods can be experienced every day and they coincide with business telephone peak hours. Incidental overload may occur because of exceptional situations like phone-in television and radio shows. A classical example of incidental overload happens every year at five minutes past midnight on January 1.

The effect of overload can be disastrous. In the spring of 1988 a television show in which viewers were requested to call the studio, caused a total breakdown of the Netherlands' telephone network for one and a half hour. A similar situation occurred after the 1989 earthquake that hit Northern California. The Bay Area telephone network almost broke down, not because of damage caused by the earthquake, but because of an overload due to incoming calls. Since a considerable amount of revenue may be lost during overload, it is of great interest for telephone companies to prevent the occurrence of such situations.

The aim of this thesis is to synthesise and analyse control algorithms that guarantee a high level of service, even under overload conditions. We shall not attempt to deal with the problem of overload control in a telephone network, but we shall restrict our attention to control of isolated telephone exchanges. These exchanges or switches are usually of SPC (Stored Program Controlled) type, which means that they are special purpose computers that run an operating system designed for maintaining telephone traffic. Although the restriction to isolated switches may seem an unrealistic

simplification, clearly we can never expect good performance of the network as a whole if we cannot guarantee that its basic operating units, the exchanges, are functioning properly.

Observation of exchanges in overload situations have revealed that, if no control algorithm is used, the performance may deteriorate to a state where no connections are established. In such a situation the switch may be working at full capacity, attempting to set up connections, but the time needed to perform these actions, the set up time, is too long compared with the impatience of the subscribers. This leads to a high number of so-called abandoned call requests, i.e. attempts for connections that were initiated but never actually established. The capacity of the switch's processing power used for the abandoned call requests is wasted and this waste, combined with the load of retrying subscribers, may lead to even longer set up times, until the vicious circle ends in 100% abandonment of call requests. In such a situation the number of call requests in service should be kept bounded to keep the set up times sufficiently small. This can for instance be accomplished by denying access to new call requests when the number of requests in service is large. Such an algorithm that guarantees correct performance of an exchange, even under overload conditions, is called an overload control law.

The mathematical models for exchanges are usually taken from queueing theory. To investigate the effects of overload control algorithms one can follow two approaches. In the first approach a control law is imposed on the arrival processes of a queueing model and the performance of the controlled model is evaluated by exact or approximate analysis or simulation. This approach is called performance analysis and with it we can try to find settings of system and control parameters that guarantee good performance, even under overload conditions. Examples of this approach can be found in DOSHI AND HEFFES [29], FORYS [36], SZE [96] and TRAN-GIA AND VAN HOORN [99]. Although there exists a vast amount of literature on performance analysis of queueing networks (see for instance DISNEY AND KÖNIG [28], KLEINROCK [53, 54], LAVENBERG [59] and SCHWARTZ [84]), few results have been obtained for queueing models with controlled arrival processes.

The second approach to the problem of overload control is based on optimal stochastic control. Here a control and reward structure is imposed on a queueing model and an optimisation criterion is formulated that reflects desirable behaviour. The problem then is to find an optimal control law, i.e. a control law that satisfies the criterion. Examples of this approach are found in SCHOUTE [80, 82, 83]. In the stochastic control approach it is important that one can deduce structural properties of the optimal control law. Such properties may be more valuable than specific numerical solutions, since they can provide insight into how a queueing system should be controlled. The derivation of these properties is called the synthesis problem of optimal control. Although a number of significant results have been obtained in this area (for an overview see EPHREMIDES AND VERDÚ [33] and STIDHAM [91]), these results are confined to queueing systems with a state space of small dimensions. With the growing complexity of today's communication systems there is an increasing need for new methodologies to derive these structural properties in complex queueing networks.

In this thesis we shall discuss several models for telephone switches and we present synthesis and analysis of overload control algorithms. The two approaches that were sketched above — performance analysis and optimal stochastic control — shall be

used in this investigation. The organisation of this thesis is as follows. In Chapter 2 we describe the functioning of an SPC exchange. We depict the modular units of a switch and we describe which actions have to be performed to set up a connection. We discuss the load on the switch's processor caused by these operations and the effects of a temporary demand for service that exceeds the capacity of the exchange. We conclude the chapter with a discussion of several aspects of control of the exchange during a period of overload.

In Chapter 3 we present the mathematical framework for the remaining chapters of this thesis. The first part of the chapter deals with the performance analysis of queueing networks with controlled arrival and departure processes. We present new sufficient conditions to have a product form equilibrium distribution in such a network. The existence of a closed form formula for several performance measures, inherent to the product form, can assist us in an efficient analysis of such networks. The second part of Chapter 3 deals with optimal stochastic control of queueing networks. We shall show how Markov Decision Processes can be used for the analysis and synthesis of control algorithms for queueing networks.

These preliminaries are put to work in Chapter 4 where an M/M/c queueing model with so-called impatient customers is considered for call processing in an SPC exchange. This simple queue with impatient customers is shown to be a good model for a switch, since it exhibits the typical overload behaviour of a real exchange when no control algorithm is used. We also show that an elementary control algorithm can be implemented that guarantees good performance under all circumstances. The main part of the investigation in this chapter is carried out with the optimal stochastic control approach that was presented in Chapter 3.

In Chapter 5 the two approaches that were sketched in Chapter 3, viz. performance analysis and stochastic control, are combined to analyse a detailed Processor Sharing queueing model for an exchange. An important characteristic of the queueing model is that we incorporate the effects of both call requests and operator tasks. Operator tasks are issued by the operator of the exchange and, although they are not directly relevant to call processing, are essential for proper operation of an SPC switch. We discuss two classes of control algorithms. For one class we solve a constrained optimisation problem and for both classes we present an analysis of their performance.

This thesis is concluded with some remarks and suggestions for future research in Chapter 6. Two appendices are included for proofs of theorems and lemma's in Chapters 3 and 4, respectively.

2

SPC Telephone Exchanges

2.1 Introduction

During the last forty years telephone equipment has gone through the same technological revolution as computer equipment. While the earliest telephone exchanges were operated manually by human operators, most modern *Stored Program Controlled (SPC)*[1] exchanges are powerful digital computers running an operating system that is dedicated to maintaining telephone communication. The technology that is used in telephone exchanges is as advanced as the technology in computer systems. We see for instance an increasing use of fiber optics and distributed system designs.

The analogy between telephone and computer equipment is not limited to the hardware and software technologies used in their design, but we also encounter the same problems of measurement, analysis and prediction of their performance in a real life operational environment. Just as in computer equipment, most of the resources in an SPC are expensive and thus scarce. It is therefore important to determine the capacity of resources needed under nominal operating conditions. Since the demand of service of the exchange fluctuates, this capacity is likely to be insufficient for all situations, but it is an economically justifiable choice. It appears, however, that some control action is needed to guarantee a certain service level of the exchange in overload, i.e. when the demand for service exceeds the resources' capacity. Observations (cf. Forys [36]) have revealed that exchanges under overload conditions exhibit a sharp decrease in the effective throughput to unacceptably low levels, unless some form of overload control is implemented.

In this chapter we describe the operation of an SPC exchange and point out some of the causes of the SPC's typical overload behaviour. In Section 2.2 we describe its design as a dedicated digital computer. We show the analogy with ordinary computers plus some of the differences. Section 2.3 deals with the characterisation of overload.

[1] In the remainder of this thesis we shall use the abbreviation SPC for both the term Stored Program Controlled and the exchange itself.

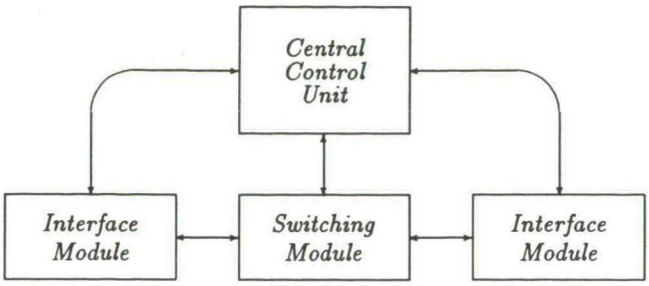

FIGURE 2.1. An SPC exchange.

We discuss congestion and point out the main cause of the SPC's typical overload behaviour. In Section 2.4 we discuss the aspect of controlling the exchange's operation under overload. This control is needed to prevent the deterioration of the SPC's call handling capacity. An important issue is the implementation of the control.

2.2 Description of an SPC

2.2.1 The modules and functions of an SPC

As we mentioned in the introduction of this chapter, an SPC is a computer that runs a dedicated operating system. The basic setup of an SPC is depicted in Figure 2.1. It comprises a *Central Control Unit (CCU)*, a *Switching Module (SM)* and one or more *Interface Modules (IM)*.

The CCU consists of one or more processors that handle tasks and messages. Messages may come from the connected modules or from tasks in the processor, and they can generate tasks themselves. A task is a set of operations that have to be performed by the processors, and it may generate messages for a module or new messages for the CCU (and thus new tasks). The processors work in a *multiprogramming* or *time sharing* fashion, i.e. they may work on several tasks simultaneously.

We can make several distinctions in the variety of tasks, e.g. we can distinguish between realtime and non-realtime tasks, or between tasks that require immediate actions and those that can be deferred for some time. We can interpret this distinction as different requirements in the timing between messages and tasks. Realtime tasks require a series of operations and messages to be performed and issued shortly after one another. Tasks that require immediate actions are similar to realtime tasks, but usually require only one quick response to a message. These different timing requirements are implemented using priority and interrupt schemes. For instance tasks that handle faults that require immediate actions are triggered through interrupts, while administration, which can be deferred for some time, is performed at a low priority.

The IM's connect the actual switching part of the exchange to the incoming and outgoing trunks. Most of the electronic interface equipment is therefore stored in the IM's. Examples are line circuit connectors and different kinds of senders and receivers, like touch tone receivers that are used to translate touch tone signals into digits.

The SM is the real switching part of the exchange where the connections are switched and routed. In old exchanges the connections were made by electric relays. Since physical circuits were used to build connections, this implementation was

referred to as *circuit switching*. In modern exchanges the connections are present only at a virtual logical level — different connections may share a physical connection — and one usually speaks of *virtual circuit switching*.

In the earliest SPC exchanges the main processing was done in the CCU, while the SM and IM's could be considered passive modules. The IM only provided an electrical interface to the outside trunks and the connections in the SM were made through relays that were set by the CCU. With microprocessors becoming cheaper modern SPC's have a large part of the processing done in the SM and IM's. For instance, generating dial tones, pulse and dial tone detection and digit analysis are now mostly performed in the IM's, while the SM has evolved from a classical circuit switch to a virtual circuit switch, where switching is done by time-multiplexing data packets.

2.2.2 The call handling procedure

When a subscriber lifts the receiver off-hook to request a connection to another subscriber, a set of actions has to be performed at the exchange to establish this connection. These actions can be grouped into a number of procedures, that describe the actions at a more global level. We shall discuss the procedures that are performed to connect two subscribers for a *Plain Ordinary Telephone Service (POTS)* call. The procedures issue messages or signals that generate new tasks in the CCU or one of the modules (cf. SCHWARTZ [84]). In the following description the procedures' names are printed in *italics* and the signals in ***slanted typewriter*** style (see also Figure 2.2). We adopt the usual terminology to call the subscriber who requests a connection the A-subscriber, while we refer to the addressed party as the B-subscriber.

The hardware at an IM performs a constant scanning of the incoming trunks to detect changes. Changes can occur due to either a subscriber lifting the receiver off-hook or a message coming in from another exchange. If the change is caused by a subscriber, a ***connect*** signal is issued to the CCU. This procedure is called *seizure*. If the CCU decides to handle this call request, the IM is notified and replies with a ***dial tone*** signal to indicate that the equipment is ready for dialing. The A-subscriber can now transmit the address of the called party either by ***dial pulsing*** or ***touch tone signalling***.

If the connection request comes from another exchange, then the procedures in the preceding paragraph are slightly different. In this case a ***connect*** or ***seizure*** signal is detected on an incoming trunk. If the CCU decides to handle the call request, the originating exchange is notified and requested to transmit the called party's address.

After the CCU has analysed the digits, it is determined whether the call is *outgoing* or *terminating*. If the addressed subscriber is connected to the same exchange, the call is said to be terminating and the next procedure, *waking up*, can be started. If the addressed party cannot be reached directly, the CCU determines the appropriate outgoing trunk, seizes it and sends a ***connect*** or ***seizure*** signal to the next exchange. This signal requests service and holds the connection until the call is released. The procedure is called *forward signalling* and it can propagate through additional exchanges along the path to the called party. At the destination exchange, i.e. the one to which the B-subscriber is connected, on reception of this signal, the *waking up* procedure is started and a ***ringing*** signal is sent to the called party. The destination exchange also issues an ***audible ringing*** signal, back to the calling party, to indicate that ringing has begun.

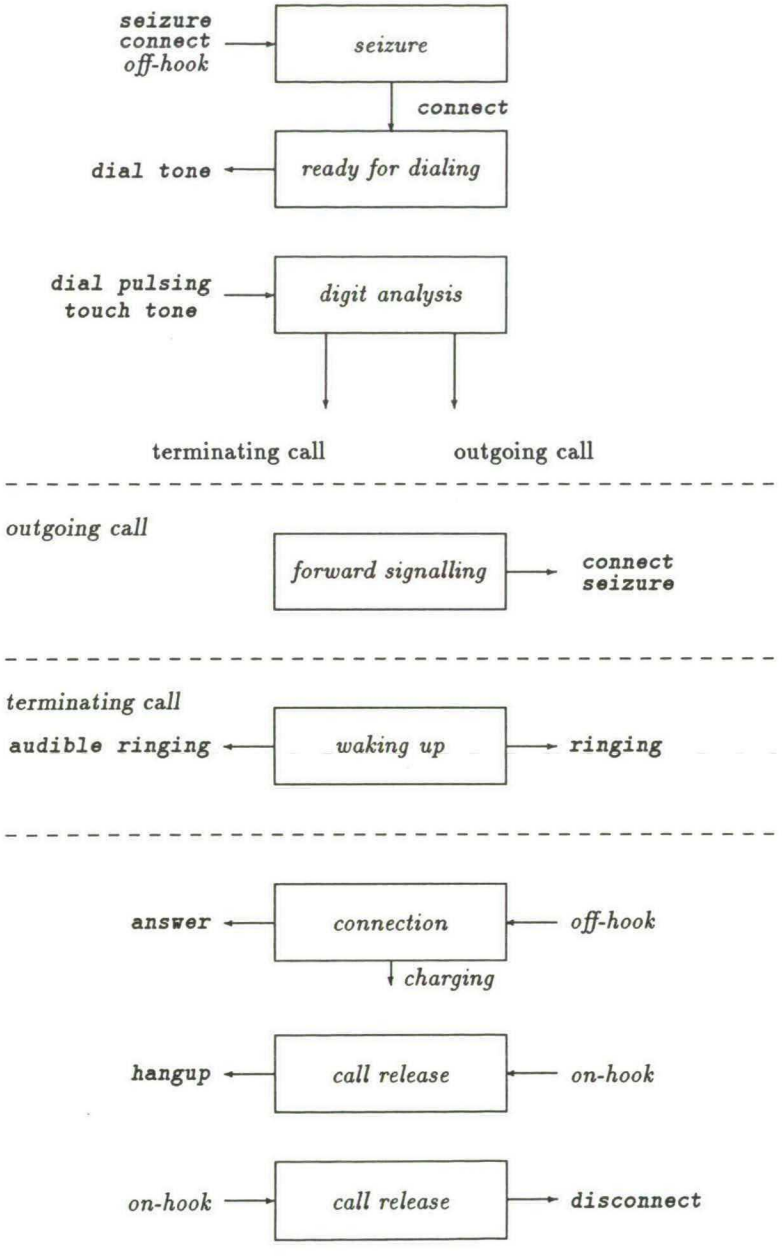

FIGURE 2.2. Global procedures and signals.

When the called party answers with an off-hook signal, an **answer** message is sent to the calling party. This signal starts the *connection* procedure that disconnects ringing, establishes a circuit between the A- and the B-subscriber and starts charging. Alternatively, a **busy** signal can be sent back to the calling party, to indicate that either the called line or the network is busy. Finally, when an on-hook condition is detected at either subscriber, the call is completed. If the A-subscriber completes, a **disconnect** signal is sent to the other party. If the B-subscriber terminates the connection, a **hangup** signal is returned to the calling party.

The architecture of the SPC determines which module is taking care of each of these procedures. In a non-distributed exchange, the CCU's processor is responsible for all procedures and the SM and IM's act merely as hardware resources. This centralisation imposes a large and complex load on the processor and can cause problems for realtime tasks. For instance the counting of the dial pulses of a rotary dial telephone requires accurate timing and has to be done at a high priority to prevent pulses from being missed. When the exchange is operating at a high load, it is not unlikely that the realtime tasks go wrong because the processor is busy working on other tasks.

Recently there has been a tendency to design SPC exchanges with distributed processing. The motivation for this is twofold. First it relieves the CCU of tasks that can better be performed at the peripheral modules. The IM, for instance, is a more appropriate place to perform dial pulse counting than the CCU, since that is also the module where the touch tone detection equipment is located. The second motivation is that it is easier to build small, fast processors for a restricted set of procedures than to build a large and fast processor that has to perform a large variety of tasks. The advantages of distributed systems become even more apparent as larger exchanges are being built and more services, like call forwarding and call waiting, are being offered. All the elementary call procedures can then be performed at the modules' processors and the CCU processor can restrict itself to central control of the modules and the advanced call services. A distributed or modular setup also allows for easier maintenance and upgrading. The AT&T No. 5 ESS (electronic switching system) is an example of such a distributed exchange (cf. DUNCAN AND HUEN [30]). It consists of interface modules that take care of most of the call processing functions, while the CCU directs inter-module routing through the time-multiplexed SM. For this SPC the description of the call procedures applies to the operations at the IM's processors rather than the CCU's.

Another recent development is the introduction of *Common Channel Signalling*. Up to a few years ago an exchange had to seize a trunk when it wanted to send the connect signal to the next exchange. All the messages that were sent back and forth between the originating and terminating exchanges were sent over these trunks. In a Common Channel Signalling network all messages to set up a connection are sent over one high-speed channel. Not only does this lead to a more efficient use of the trunks, it also provides a fast communication channel between exchanges, e.g. to send control information about congestion.

It must be apparent that the weight of the different call procedures, i.e. the CPU time required to perform them, is strongly dependent on the system design. As a rough indication one can say that the *seizure* procedure takes about 25% of the total load of a call request as does *dialing* together with *digit analysis*. Setting up the connection (*forward signalling, waking up*, and *connection*) takes about 40% and *call release* 10%.

2.3 Characterisation of overload

Before a new exchange is installed, a study is made of the volume and type of traffic that the exchange will be expected to handle. Traffic volume is defined as the mean number of call requests per second (the call arrival rate) times the mean call duration in seconds and the appropriate unit is called *Erlang*. When an exchange is offered 0.5 call requests per second and the mean call duration is 60 seconds, then the offered load is equal to 30 Erlang.

An SPC is configured to make its call handling capacity large enough to deal with call requests under nominal load. Since traffic fluctuates, SPC's are usually dimensioned to use 80% of their design capacity when offered a (constant) nominal load. In this way enough room is left to cope with small short term fluctuations. The capacity for nominal conditions is determined in the following way. Most telephone companies use nominal specifications like timing requirements to relate traffic volume to design capacity. In the United States' telephone network, for instance, one of the requirements is that the probability of *dial tone delay* — the delay between off-hook and the generation of a dial tone — exceeding 3 seconds is smaller than 0.01. Similarly *connect delay* — the delay between the B-subscriber's off-hook and the end of the connection procedure — is not allowed to exceed 0.5 seconds with a probability larger than 0.01.

With these guidelines it is possible to determine the design capacity for an exchange. Under a slightly varying nominal load it is then guaranteed that all call requests can be handled and that delays are sufficiently small. In practice, however, one can experience large deviations from the nominal load. If the actual load exceeds the design capacity, then we speak of *overload*. One can distinguish between *structural* and *incidental overload*. Structural overload can be experienced everyday at roughly the same time of day and coincides with business telephone peak hours. Incidental overload is caused by exceptional situations and happens less frequently than structural overload. Examples are the load at 00.05 AM on January 1 or the load caused by phone-in TV and radio shows.

During these periods of overload it is not unusual to have a temporary load four times as large as the nominal one. It is not economically feasible to design the capacity large enough to deal with overload and at the same time satisfy the timing requirements, since too much of the capacity would then remain unused under nominal load. When, on the other hand, an exchange that is designed for nominal load, tries to handle overload traffic by accepting all offered call requests, the dial tone and connect delays will become quite large.

This violation of the nominal specifications manifests itself in a sharp decrease of the *effective call handling rate (ECR)*. The ECR is defined as the mean number of completed call requests per second, where a call request is said to be *completed* if it ends with a successful connection. This quantity is also referred to as *goodput*, the throughput of 'good' or 'successful' call requests. Even under nominal load the ECR is smaller than the call arrival rate due to busy signals, but this loss is negligible. When the exchange is overloaded, however, the dial tone delay may become so large that a subscriber abandons his call request by putting the receiver back on-hook before a dial tone is issued. This action may be detected quite late at the exchange, even after a superfluous dial tone had been sent and the exchange had spent some time scanning the line for incoming digits. Another reason why a subscriber might want to cancel

his call request is when it takes too long to generate the ringing signal. The situation can become even worse, if the subscriber becomes impatient and presses the receiver's hook a couple of times. All these changes are detected and stored and it can take quite a while before the exchange realises that it is handling a number of call requests for one subscriber and only one request is still valid at the time. All these actions lead to a situation where the SPC is wasting a part of its capacity on operations that will not lead to a completed connection. If no precautions are taken, this waste can lead to even longer delays and more capacity wasted. In the ultimate situation the exchange might end up in not completing any call requests at all. This phenomenon of a sharp increase of delays and decrease of the ECR is called *congestion*. This behaviour motivates that some means of control is needed to prevent congestion of an SPC in an overloaded environment.

2.4 Aspects of overload control

The mechanisms that are implemented in SPC's to establish correct operation during overload conditions are called *overload control* and they form the main topic of this thesis. An overload control algorithm is implemented as a part of the computer program that makes up the operating system of the SPC. Although problems of overload control are not confined to SPC's but occur in communication networks in general, we shall restrict our attention to telephone switches. In this section we discuss some aspects of overload control, specifically control objectives, implementation, information patterns, robustness and adaptive control.

The first step in the design of an overload control mechanism for an SPC is to develop a mathematical model for the dynamic behaviour of the exchange. Just as in the modelling of ordinary computer systems queueing models seem an appropriate choice. In literature a variety of queueing models has been proposed, e.g. M/M/1 queues in Doshi and Heffes [29], M/G/1 queues in Forys [36], Sze [96], hierarchical queueing systems in Schoute [81, 82] and priority queueing models in Karlander [48].

The next step is to formulate a *control objective* that in some form resembles the real life requirement: make as many successful subscriber connections as possible. As we have seen in Section 2.3 one of the main causes of call requests becoming unsuccessful is the impatience of subscribers. With this in mind there are two intuitively appealing choices for the modelling of successful subscriber connections, and hence of the control objective.

In the first approach impatience of subscribers is explicitly accounted for in the queueing model and the control objective is to maximise the effective call handling rate or goodput. In general, however, analysis of queueing systems with impatient and thus potentially unsuccessful customers is quite difficult and in literature this approach is limited to queueing models consisting of one queue (cf. Doshi and Heffes [29], Fayolle and Brun [35], Forys [36], Sze [96] and Tran-Gia and van Hoorn [99]).

In the second approach impatience of customers is implicitly accounted for. Usually a call request becomes unsuccessful when the time that has elapsed since it was initiated — the sojourn time — exceeds some deterministic or random value. Due to this relation between impatience and sojourn times, one can try to keep sojourn times bounded to reduce the performance loss caused by unsuccessful call requests. This leads to a control objective formulated as a constrained optimisation problem: maximise the throughput of (both successful and unsuccessful) call requests while

keeping their sojourn times bounded. The constraint may be formulated for example as a bound on the mean sojourn time or as a bound on the sojourn time distribution. Examples of this approach can be found in SCHOUTE [82] and ROBERTAZZI AND LAZAR [74], the latter of which deals with the related problem of flow control in data communication. We present examples of both approaches in Chapters 4 and 5.

The next step is to determine in which manner the control is to be implemented. In literature we can distinguish two approaches, viz. *access control* and *scheduling control*. In access control the implementation is an algorithm that allows rejection and admittance of customers. Rejected customers are not queued and assumed lost. An example of this approach is presented in SCHOUTE [82]. Scheduling control may be viewed as an supplement of access control. Newly arriving customers may be admitted to the SPC immediately at the moment of arrival according to a control algorithm. Customers who are not admitted immediately, are put in a finite buffer until a call request in service completes and a request from the buffer can be admitted. The scheduling control determines how this buffer is emptied when a new call request can be admitted to the SPC. In DOSHI AND HEFFES [29] five disciplines are compared, viz. combinations of *First Come First Served (FCFS)* or *Last Come First Served (LCFS)* with *push-out* (the oldest customer is removed when the buffer is full), *blocking* (new call requests are rejected when the buffer is full) or *time-out* (call requests are removed from the buffer when their sojourn time exceeds some threshold). In FORYS [36] a LCFS discipline is proposed with a modification to reward customers that have continued to wait beyond a specified threshold by putting them at the head of the queue at that time. In SCHWARTZ [84] both approaches, viz. access control and FCFS scheduling control are compared.

An important issue in the implementation of the control algorithm concerns its complexity. As we noted before, the overload control is implemented as a part of the SPC's operating system. One can design a very sophisticated algorithm to perform the control, but if the algorithm requires a considerable amount of processing power, that has to delivered by the SPC's processor, the cost of the overload control might exceed its benefits. With the advent of distributed processing and decreasing costs of microprocessors this requirement is likely to become less important in the future, since dedicated processors may be used to solely perform control. Nevertheless complicated control algorithms, even when run on a dedicated processor, can be less desirable because of the information that is required to perform the control actions.

This brings us to the problem of deciding what information should be available for the control algorithm to decide when to control access to the exchange. For any particular queueing model of an exchange, the algorithm would ideally have complete state information, i.e. the state of the underlying Markov process. Depending on the complexity of the queueing model problems may arise when trying to make this information available. The desired information may, for instance, either not be available or at the cost of a high processing effort only. Sometimes only part of the information is available. In a distributed environment complete up-to-date state information may involve a high volume of message passing between processors that could reduce the benefit of the control. Information about remote sites or modules may only be available periodically.

Once a model and a control objective have been formulated, we have to find a control law that satisfies this control objective. This problem is called *control synthesis*. It is

addressed to with methods from optimal stochastic control. The nature of overload control points to an environment where a number of parameters may change in time. As an example we have already mentioned the call arrival rate, that may change from about zero during the night up to four times the design capacity during peak hours. The problem of the availability of information is twofold in this situation. First, it is not clear beforehand whether and how the dynamics of the parameters are to be modeled. Second, the actual values of the parameters are generally not known.

There are two ways to tackle this problem. First we can try to design an implementation of an overload control mechanism that does not depend on the actual parameter values. Such a control is called *robust*. An alternative approach is to design an *adaptive controller*, where the implementation of the controller varies with the changing parameter values. In adaptive control itself we can distinguish two different approaches. In the first approach one tries to estimate the unknown parameters on the basis of observations of the queueing system and use the estimates as if they represent the real system and adjust the control accordingly. In the second approach the control mechanism is adjusted on the basis of the observations without making estimates of the parameters. Examples of this last approach are presented in BERGER [9] and DAISENBERGER *et al.* [26].

In this chapter we have given an informal introduction to the problem of overload control of SPC telephone switches. We have described the basic operations of an SPC, the main cause of its performance deterioration during overload and some aspects of the implementation of a control algorithm to prevent this behaviour.

3

Performance Analysis and Control of Queueing Systems

3.1 Introduction

A telephone exchange can be characterised as a collection of resources that provide services to customers. Since the resources are limited both in capacity and number, customers will contend for the available resources and waiting lines will be formed. This observation justifies the choice of queueing systems as an appropriate model for the exchange's dynamic behaviour.

Queueing systems were introduced in Erlang [34] and Engset [31] as modelling tools in telephone theory. The fundamental entities in their models for the dynamics of trunk occupation were *servers* and *customers*. A customer represented a subscriber requesting a trunk for a connection. Each trunk was represented by a server in the queueing model and the time that a server needed to serve a customer, represented the holding time of the connection. If a customer arrived at the servers, i.e. requested a connection, and no trunk was available, then the customer left the queueing system without service and did not return. Since there was no possibility for customers to wait, these queueing systems were called *loss systems*.

In the course of the last fifty years models of queueing systems evolved from these fairly simple loss systems to complex queueing networks. The extensions that were introduced included waiting spaces to store waiting customers, customer types to cope with customers having different characteristics, and service disciplines to prescribe the order in which customers had to be served. In Jackson [46] *queueing networks* were introduced to model manufacturing systems: servers were grouped into a number of stations and a routing mechanism was used to describe how customers, after completion of service at one station, would jump to the next station.

In the 1960's queueing networks proved to be valuable in the modelling of multi-user computer systems. From real life observations it had appeared that, except under very light loads, the performance of such systems was strongly affected by the queueing delays encountered by the users. Queueing network models could be used to predict the performance of these systems and give indications about the performance sensitivity

with respect to the various system parameters (see e.g. LAVENBERG [59]).

Since modern telephone exchanges are typically computer systems, queueing networks have been used extensively as models in SPC performance analysis. In models for computer systems and telephone exchanges customers are alternatively active and idle, i.e. either requesting service from one of the resources of the system or doing nothing. There is, however, a fundamental difference between queueing network models for modern time-sharing computer systems and telephone exchanges. In models for time-sharing computer systems, where the customers represent the interactive users of the system, a customer is called active for instance when he is using the CPU or disk and he is called idle when he is evaluating output from a program and thinking of new input. A customer in SPC models, representing a subscriber, is active between the moments of off-hook and on-hook, and idle otherwise. Since a typical interactive computer session comprises a consecutive series of active and idle periods, the idle periods are explicitly accounted for in the model. For this reason the number of customers in computer queueing network models is usually held constant and the networks are called *closed*. A typical "telephone session" usually consists of exactly one active period. Therefore the idle period is not explicitly modelled and the arrival of a customer – the start of a request for service – is modelled as being generated by an infinite source of customers. These models are referred to as being *open*.

Once a model has been chosen, we can address the problem of controlling its behaviour under overload in two ways, which shall be discussed in the remaining two sections of this chapter. In the first approach a control is imposed upon the model and the performance of the controlled model is evaluated. This approach is standard *performance analysis* for which modelling and analysis techniques are widely available in the literature (see e.g. DISNEY AND KÖNIG [28], KLEINROCK [53, 54] and SCHWARTZ [84] for an overview). With exact or approximate analysis or by simulation we can predict the values of performance measures like throughputs (e.g. the mean number of completed call requests per time unit) and sojourn times (e.g. the time to set up a connection).

There exists a class of queueing networks that are very appropriate for modelling of computer systems. This is the class of product form queueing networks, so called because of the product form of their steady state distribution. This product form allows an easy evaluation of the performance metrics of these networks. They can be used to model both open and closed queueing networks (plus mixtures of these) and they are therefore suitable candidates for models of an SPC. In the literature on product form networks, however, relatively little attention has been paid to networks with access control on the arrival processes. In Section 3.2 we shall discuss an extension to the class of product form queueing networks that allows arriving customers to be blocked or rejected. These results shall be used in Chapter 5 for a product form queueing model of an SPC.

In the second approach to the problem of overload control, which is based on *stochastic control*, a control and reward structure is imposed on the queueing model and an optimality criterion is formulated that in some form reflects a desirable behaviour. An example of such a criterion is 'handle as many call requests as possible while keeping the time to set up a connection bounded'. Techniques like dynamic programming and value or policy iteration (see e.g. ROSS [76, 77] and TIJMS [97]) are available to solve for the optimal control, i.e. the control that satisfies the optimality criterion.

In Section 3.3 we shall discuss some aspects of optimal control of queueing systems. The results shall be used in Chapter 4 to deal with overload control of a queue with impatient customers.

3.2 Product form networks with controlled arrivals

In the analysis of queueing network models an important role is played by the equilibrium distribution of the underlying (Semi-)Markov process. Under some ergodicity conditions this distribution can be computed as the solution of a set of linear equations. For most models of queueing networks the size of this set of equations is often too large to allow a straightforward computation of its solution. There exists a class of queueing networks for which the equilibrium distribution can be written explicitly as a product of terms for each station in the network. In this product each term represents the equilibrium distribution of the station viewed in isolation. This is the class of *product form queueing networks*. In open networks the terms for all stations can be combined into a single product formula that represents the equilibrium distribution of the network as a whole. In closed networks this can be accomplished too, provided one accounts for the fixed sizes of the customer populations.

An important advantage of product form networks, apart from the closed form expression for the equilibrium distribution, is the fact that performance measures can be evaluated in a relatively simple and efficient way. In open networks the steady state distribution has a form that allows the computation of performance metrics directly. For measures such as mean queue lengths and waiting times closed form expressions can be derived that can be computed in a straightforward manner. In closed queueing networks the situation, however, is more complicated. This is caused by the fact that in combining the terms of each station's distribution into a product form for the network's distribution one has to exclude network states that do not have the correct (fixed) customer class population. This introduces a normalisation constant in the product form to guarantee that the product form represents a proper distribution (i.e. all probabilities sum up to one). For evaluation of the performance of a product form network one therefore may have to compute the value of this constant. A number of algorithms have been developed to deal with this problem. These algorithms can be divided in two groups. The first group contains the convolution-like algorithms (cf. e.g. Buzen [19], Reiser and Kobayashi [72], Conway and Georganas [24, 25]) that combine the computation of both the normalisation constant and several performance measures into a convolution on the network's state space. The second group of so-called mean-value-analysis algorithms (cf. e.g. Reiser [71], and Reiser and Lavenberg [73]) completely bypasses the computation of the normalisation constant, but allows only the evaluation of first moments of the performance metrics.

In this section we discuss a method to introduce control of arrivals in a queueing network and we give conditions under which the equilibrium distribution has a product form. It has been published in a different form in de Waal and van Dijk [101].

3.2.1 Introduction

Product form queueing networks were first introduced in Jackson [46]. He considered single-chain networks with Poisson input and exponential service time distributions. Gordon and Newell established product form results for closed queueing networks with

exponential service (cf. GORDON AND NEWELL [39]). Closely related to the existence of a product form equilibrium distribution is the *insensitivity property*. By insensitivity is meant that the equilibrium distribution depends only on the means of the service time distributions and not on higher moments. It appeared that both the product form and the insensitivity could be explained by the notion of *local* or *partial balance* (cf. BASKETT *et al.* [6], CHANDY *et al.* [20], CHANDY AND MARTIN [21], HORDIJK AND VAN DIJK [44], KELLY [50], SCHASSBERGER [79], WHITTLE [106, 107]). Local balance means that there exists balance in probability flows on subsets of the state space.

Although the queueing network models as described in BASKETT *et al.* [6] — usually called *BCMP-networks* as a reference to the authors' last names — have been used extensively in the context of telecommunications, computer performance modelling and flexible manufacturing, they have some serious drawbacks that limit application in a large class of real life systems. In general blocking is not allowed and, in open networks, arrival rates may depend on the network occupancy only in an elementary way. It is for example not allowed to let the arrival rate for a customer class depend on the occupancy of another class. A first extension in this direction was presented in LAM [58]. He provided sufficient conditions for an open queueing network to prohibit arrivals or departures depending on the momentary multiclass network occupancy. However, inputs were assumed to originate from Poisson sources and arrivals and departures could only be rejected or admitted without randomisation. Related results can be found in KAUFMAN [49] and FOSCHINI AND GOPINATH [37] where a single station with Poisson input is allowed to restrict the set of feasible states to a coordinate convex set by rejecting arriving customers.

In BURMAN *et al.* [17], HORDIJK AND VAN DIJK [42], KELLY [50], PITTEL [69] Poissonian arrivals can be blocked due to finite capacity constraints in single stations under the restrictive condition of a reversible routing. In HORDIJK AND VAN DIJK [44] a number of extensions to some special non-reversible examples are given. Blocking depending on the total configuration of a group of stations, however, was hereby not involved. Extensions in this direction can be found in VAN DIJK [27], KRZESINSKI [56], SERFOZO [89] and TOWSLEY [98]. In [56] and [98] customers can only be blocked when jumping inside a cluster, however, and not upon entering and leaving a cluster (cf. [27, Section 4.3 (iii)]). The results in [27] and [89] are limited to single class exponential structures.

This section will study multiclass non-exponential networks with both arrival and departure blocking depending on the total customer class occupancies and with arrivals generated by another non-exponential network. An extension of the standard insensitivity product form results for BCMP-networks will be obtained. As a particular extension we discuss open and mixed open-closed networks. Some further extensions, such as to different blocking protocols, class changes and network dependent service disciplines will be discussed in Appendix A.

3.2.2 Main results

3.2.2.1 Description of the model

In this subsection we give the description of the queueing network model. We let $\mathbb{N}$ denote the set of natural numbers and $\mathbb{N}_k = \{1, \ldots, k\}$. Consider a network of N stations. The stations are grouped into two clusters named C_1 and C_2. Stations

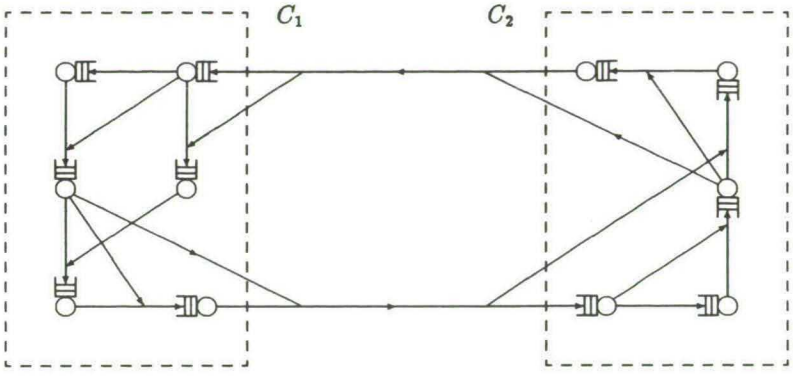

FIGURE 3.1. A network with two clusters of queueing stations.

in cluster C_1 are numbered $n = 1, \ldots, N_1$ and stations in C_2 are numbered $n = N_1 + 1, \ldots, N$. We let $\mathcal{N}$ denote the set of stations $C_1 \cup C_2$. An example of a network with two clusters is depicted in Figure 3.1.

Following KELLY [50] the service disciplines at the different nodes are described by three functions f_n, ϕ_n and δ_n, that have the following interpretation.

$f_n(k)$ the speed of the server at the n-th station when k customers are present,

$\phi_n(k,i)$ the fraction of the total service speed that is awarded to the customer in the i-th position at station n when k customers are present,

$\delta_n(k,i)$ the probability that a customer arriving at station n is placed at position i when k customers are present just before the arrival.

We assume that the shift protocol is used, i.e. if a new customer is placed at position i, then the customers at positions $i, \ldots, k$ shift to positions $i+1, \ldots, k+1$, and if a customer departs from position i, then these customers shift to positions $i, \ldots, k-1$. We also assume the buffers at all stations to be infinite. Note that from the definition of ϕ and δ we have

$$\sum_{i=1}^{k} \phi_n(k,i) = \sum_{i=1}^{k} \delta_n(k-1,i) = 1. \tag{3.1}$$

DEFINITION 3.2.1 (SYMMETRIC QUEUES)

(i) Queue n is symmetric *if for all i and k*

$$\phi_n(k,i) = \delta_n(k-1,i). \tag{3.2}$$

(ii) The set of stations with symmetric queues is denoted as $\mathcal{S} \subset \mathcal{N}$.

If a queue is not symmetric, then we assume that it operates under the *First Come First Served (FCFS)* queueing discipline, i.e. $\phi(k,i) = 1$ iff $i = 1$ and $\delta(k,i) = 1$ iff $i = k+1$.

In the network we distinguish K different classes of customers. The network is closed and the total number of class k customers is equal to M^k, $k = 1,\ldots,K$. Although network models for telephone sessions are usually open, we consider only closed networks in this section. With the technique as described in HORDIJK AND VAN DIJK [43, pages 433–435] stations in the network can be replaced by infinite sources, thus extending the results to open and mixed open-closed networks.

On leaving queue m, a customer of class k moves to a queue n with probability R^k_{mn}. Note that for any station $m \in \mathcal{N}$

$$\sum_{n\in\mathcal{N}} R^k_{mn} = 1. \tag{3.3}$$

We assume the routing matrices $[R^k_{mn}]$ to be irreducible matrices for each class k. The routing from a station in one cluster to a station in another cluster is required to take a special form. We assume that for stations $m \in C_1$ and $n \in C_2$ the routing probability R^k_{mn} can be written as

$$R^k_{mn} = R^k_{m0}R^k_{0n}, \tag{3.4}$$

where

$$R^k_{m0} = 1 - \sum_{l\in C_1} R^k_{ml}, \tag{3.5}$$

and

$$1 = \sum_{n\in C_2} R^k_{0n}. \tag{3.6}$$

Similarly we assume for the routing from $m \in C_2$ to $n \in C_1$ that (3.4) holds and

$$R^k_{m0} = 1 - \sum_{l\in C_2} R^k_{ml} \tag{3.7}$$

and

$$1 = \sum_{n\in C_1} R^k_{0n}. \tag{3.8}$$

One can view this as if the jump from one cluster to another is taking place through a dummy node labeled 0, that has an infinite service speed. At this node a customer does not experience any delay and for the routing into the next cluster all information about the station that the customer came from is lost. In words this means that the event of routing of a customer from one cluster into another is independent of the originating station. Define the *visiting ratio* θ^k_n of class k to station n as the solution of the set of equations

$$\theta^k_n = \sum_{m\in\mathcal{N}} \theta^k_m R^k_{mn}, \qquad n = 1,\ldots,N, \tag{3.9}$$

with the normalisation condition $\sum_{n=1}^{N} \theta_n^k = 1$, $k = 1, \ldots, K$. Since the routing matrices are assumed to be irreducible, the solution to this set of equations exists and is unique due to the normalisation condition. By restriction (3.4) on the routing probabilities from one cluster to another, we have the following proposition.

PROPOSITION 3.2.2 *For all* $k = 1, \ldots, K$

$$\sum_{n \in C_1} \theta_n^k R_{n0}^k = \sum_{n \in C_2} \theta_n^k R_{n0}^k. \tag{3.10}$$

This result is not surprising, since it simply states that the number of jumps from C_1 to C_2 in a routing cycle through both clusters must equal the number of jumps from C_2 to C_1. One can derive Proposition 3.2.2 quite easily from (3.9) by summing over $n = 1, \ldots, N_1$:

$$\begin{aligned}
\sum_{n \in C_1} \theta_n^k - \sum_{n \in C_1} \sum_{m \in C_1} \theta_m^k R_{mn}^k &= \sum_{n \in C_1} \sum_{m \in C_2} \theta_m^k R_{m0}^k R_{0n}^k \Leftrightarrow \\
\sum_{m \in C_1} \theta_m^k \left(1 - \sum_{n \in C_1} R_{mn}^k\right) &= \sum_{m \in C_2} \theta_m^k R_{m0}^k \sum_{n \in C_1} R_{0n}^k \Leftrightarrow \\
\sum_{m \in C_1} \theta_m^k R_{m0}^k &= \sum_{m \in C_2} \theta_m^k R_{m0}^k.
\end{aligned}$$

While travelling through the network customers pass through the stations where they request service. We assume that the service demands of customers of class k at a symmetric station $n \in \mathcal{S}$ are independent and drawn from a distribution with distribution function $B_n^k(\cdot)$. Without loss of generality we assume that these distributions are absolute continuous with density functions $\beta_n^k(\cdot)$ and mean service rates μ_n^k. For each FCFS queueing station $n \in \mathcal{N} - \mathcal{S}$, the service requirements must be exponentially distributed and independent of the customer class, with mean service rate μ_n. This is a standard requirement in queueing networks to get a product form distribution.

A state of the network is represented by a vector $\mathbf{n} = ((x_{ns}), n \in \mathcal{N}, s = 1, \ldots, k_n)$, where k_n denotes the number of customers present at queue n and x_{ns} the class of the customer at position s in station n. Define a *microstate* $\mathbf{X} = ((x_{ns}, r_{ns}), n \in \mathcal{N}, s = 1, \ldots, k_n)$, where r_{ns} is the residual service requirement of the customer at the s-th position in station n. In addition we shall also use the following notations for states:

$\mathbf{X} - (n, s)$ The state that results from state $\mathbf{X}$ if we remove the customer at position s in station n.

$\mathbf{X} - (n_1, s_1) + (n_2, s_2, y)$ State $\mathbf{X}$ with the customer at position s_1 in station n_1 removed and put into position s_2 at station n_2 with a residual service requirement y.

3.2.2.2 *The invariance condition*

A *population vector* is defined as an element of $\mathbb{N}^K$. For a given state $\mathbf{X}$ we define $M_1 = (M_1^1, M_1^2, \ldots, M_1^K)$ and $M_2 = (M_2^1, M_2^2, \ldots, M_2^K)$ by

$$M_i^k = \sum_{n \in C_i} \sum_{s=1}^{k_n} 1(x_{ns} = k) \qquad i = 1, 2; \quad k = 1, \ldots, K. \tag{3.11}$$

In words the k-th component M_i^k of a population vector M_i is the total number of class k customers in cluster C_i. Since the network is closed, the sum of M_1 and M_2 must be constant. Denote this sum vector as $\mathcal{M} = \{M^1, \ldots, M^K\}$.

Next we introduce the *arrival and departure probability functions* $A^k, D^k : \mathbb{N}^{N-N_1} \to [0,1]$, that describe when jumps from one cluster to another are allowed.

DEFINITION 3.2.3 (RECIRCULATE BLOCKING PROTOCOL)

$A^k(M_2)$ *The probability that an arrival of a class k customer, $k = 1, \ldots, K$, coming from C_1, is accepted at C_2, if the C_2-population is M_2. If the customer's arrival is accepted he will route into C_2 according to the normal routing probability R^k_{0n}, $n \in C_2$, otherwise he will be rerouted into C_1 according to the routing probabilities R^k_{0n}, $n \in C_1$.*

$D^k(M_2)$ *The probability that a departure of a class k customer, $k = 1, \ldots, K$, coming from C_2, is accepted into C_1, if the C_2-population is M_2. If the customer's departure is accepted, then he will route into C_1 according to the routing probabilities R^k_{0n}, $n \in C_1$, otherwise he will be rerouted into C_2 according to the probabilities R^k_{0n}, $n \in C_2$.*

Although these definitions may look asymmetric at first sight — A^k and D^k are defined as function of the C_2 population vector M_2 — they can also be formulated as functions of M_1, since $M_1 + M_2$ is a constant vector.

With these functions we can disable certain jumps from one cluster to another, by setting either A or D for certain population vectors equal to zero. With this we restrict the queueing network without controlled arrivals and departures to a subset of its original state space. Without loss of generality we assume that this restricted state space is irreducible in the sense that it has only one set of positive recurrent states.

To get a product form stationary distribution we need to impose two conditions on A and D. The first one was originally formulated in LAM [58, page 373].

CONDITION 3.2.4 (LAM'S CONDITION)
The arrival and departure functions A^k and D^k must satisfy for all $k = 1, \ldots, K$

$$A^k(M_2) = 0 \Leftrightarrow D^k(M_2 + \mathbf{e}_k) = 0, \tag{3.12}$$

where $\mathbf{e}_k$ is the k-th unit vector.

The arrival and departure probabilities must thus be constructed such that if an arrival of a class k customer is prohibited for a certain population vector M, then departures for the same class k must also be prohibited for the population vector $M + \mathbf{e}_k$. This leads us to the definition of *paths*.

DEFINITION 3.2.5 (PATHS)
Let $\mathbf{p}$ and $\mathbf{q}$ be elements of $\mathbb{N}^K$. If there exists an $l \in \mathbb{N}$ and a sequence $\mathbf{v} : \mathbb{N}_l \to \mathbb{N}^K$, such that for all $n \in \mathbb{N}_{l-1}$ either $\mathbf{v}_{n+1} - \mathbf{v}_n = \mathbf{e}_k$ or $\mathbf{v}_{n+1} - \mathbf{v}_n = -\mathbf{e}_k$ for some $k = 1, \ldots, K$, with $\mathbf{v}_0 = \mathbf{p}$, $\mathbf{v}_l = \mathbf{q}$ and

$$\prod_{n=0}^{l-1} Z(\mathbf{v}_n, \mathbf{v}_{n+1}) > 0, \tag{3.13}$$

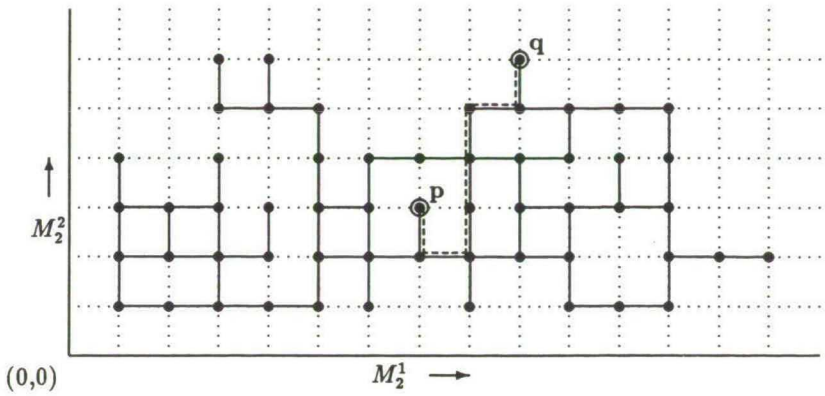

FIGURE 3.2. An example of a set of reachable population vectors and a path.

where

$$Z(\mathbf{v}_n, \mathbf{v}_{n+1}) = \begin{cases} A^k(\mathbf{v}_n), & \text{if } \mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{e}_k, \\ D^k(\mathbf{v}_{n+1}), & \text{if } \mathbf{v}_{n+1} = \mathbf{v}_n - \mathbf{e}_k, \end{cases} \tag{3.14}$$

then the sequence $\mathbf{v}$ *is called a* path *from* $\mathbf{p}$ *to* $\mathbf{q}$ *and* $\mathbf{q}$ *is said to be* reachable *from* $\mathbf{p}$.

With the definitions of the arrival and departure probabilities, there exists a path from $\mathbf{p}$ to $\mathbf{q}$ when it is possible to construct with positive probability a realisation of departures and arrivals between C_1 and C_2 such that the initial C_2-population is $\mathbf{p}$ and the terminating population is $\mathbf{q}$ (provided the routing probabilities admit this construction). Expression (3.13) is exactly the probability of this particular realisation. An example of a path for a queueing network with two customer classes is depicted in Figure 3.2. The solid points represent the set of possible population vectors in C_2 and the solid lines represent the arrival and departure transitions that are allowed with positive probability. The dashed lines show an example of a path from $\mathbf{p}$ to $\mathbf{q}$.

One can easily show that the reachability as defined in Definition 3.2.5 is an equivalence relation (the symmetric property follows from Condition 3.2.4). Since we have assumed that the restriction of arrivals and departures induces a restricted state space that is irreducible, we have exactly one equivalence class denoted as $\mathcal{E}$. We can now state the second condition we need on the arrival and departure probability functions to get a product form distribution.

CONDITION 3.2.6 (INVARIANCE CONDITION)
Let $M_0 \in \mathcal{E}$ *be the initial* C_2*-population vector, i.e. the population of* C_2 *at time* $t = 0$. *For all* $\mathbf{m} \in \mathcal{E}$ *and for all paths* $\mathbf{v}$ *from* M_0 *to* $\mathbf{m}$ *the product*

$$F(\mathbf{m}) = \prod_{n=0}^{l-1} H(\mathbf{v}_n, \mathbf{v}_{n+1}), \tag{3.15}$$

where

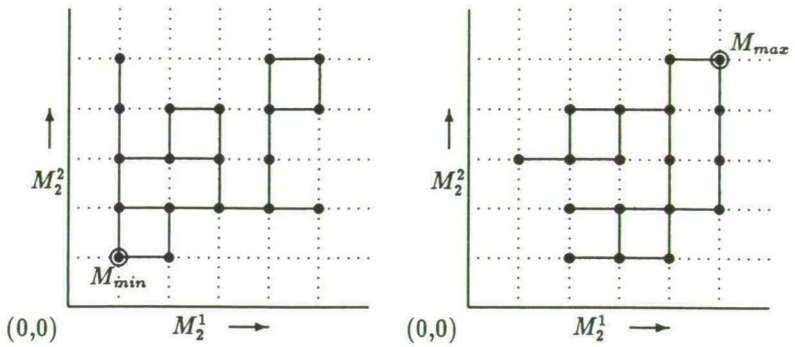

FIGURE 3.3. Examples with minimal and maximal states.

$$H(\mathbf{v}_n, \mathbf{v}_{n+1}) = \begin{cases} \dfrac{A^k(\mathbf{v}_n)}{D^k(\mathbf{v}_{n+1})}, & \text{if } \mathbf{v}_{n+1} - \mathbf{v}_n = \mathbf{e}_k, \\ \dfrac{D^k(\mathbf{v}_{n+1})}{A^k(\mathbf{v}_n)}, & \text{if } \mathbf{v}_{n+1} - v_n = -\mathbf{e}_k, \end{cases} \tag{3.16}$$

must be independent of $\mathbf{v}$.

Note that (3.15) is well-defined, since the denominators in (3.16) are always positive due to Condition 3.2.4. In words Condition 3.2.6 says that the probability to make a path from $\mathbf{p}$ to $\mathbf{q}$ must be independent of the chosen path. The expressions for $F(\cdot)$ and $H(\cdot,\cdot)$ are fairly complicated to allow for an irregular structure of the arrival and departure functions $A(\cdot)$ and $D(\cdot)$ like for example in Figure 3.2. This complexity can be avoided if there exists a *minimal state*, i.e. a state from which a path can be made to all other reachable states only by moving customers from C_1 to C_2, i.e. $\mathbf{v}_{n+1} - \mathbf{v}_n = +\mathbf{e}_k$ for all n. The product F then becomes less complicated by choosing this minimal state as the initial state M_0. A similar statement can be made if there exists a *maximal state*, a state from which all other states can be reached by a path with $\mathbf{v}_{n+1} - \mathbf{v}_n = -\mathbf{e}_k$. Examples of networks with a minimal and a maximal state are depicted in Figure 3.3.

3.2.2.3 Examples

In this section we present some examples of the broad class of networks that can be modelled as the network in Section 3.2.2.1 and that satisfy the conditions of Section 3.2.2.2. These examples and their corresponding insensitive product forms are not covered by existing results due to the non-exponential network input and randomised blocking. Related references for simpler cases will be mentioned as we proceed.

EXAMPLE 3.2.7 (COORDINATE CONVEX SETS)
Let $\mathcal{C}$ be a coordinate convex subset of $\mathbf{N}^K$, which means that $\mathbf{m} = (m_1, \ldots, m_K) \in \mathcal{C}$ implies $(\mathbf{m} - \mathbf{e}_k)^+ \in \mathcal{C}$ for all $k = 1, \ldots, K$, where $(\mathbf{m})^+ = \big((m_1)^+, \ldots, (m_K)^+\big)$ and $(m_k)^+ = \max(m_k, 0)$. If we define

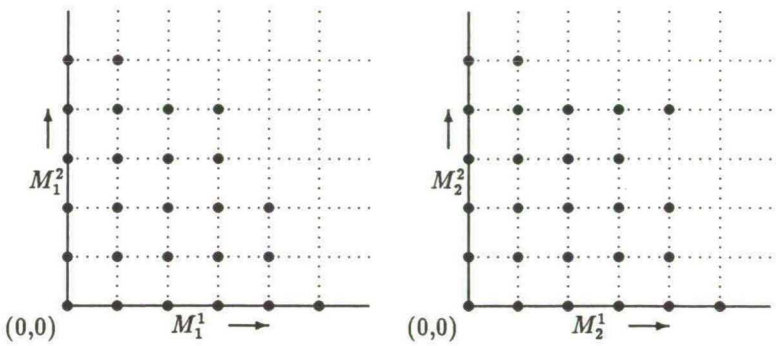

FIGURE 3.4. Example of a coordinate and a non-coordinate convex set.

$$A^k(\mathbf{m}) = 1(\mathbf{m} + \mathbf{e}_k \in \mathcal{C}), \tag{3.17}$$

$$D^k(\mathbf{m}) = 1(\mathbf{m} \in \mathcal{C}), \tag{3.18}$$

then one can show that for $M_0 = \mathbf{0}$, $\mathcal{E} = \mathcal{C}$.

For an example of a coordinate convex set and a non-coordinate convex set see Figure 3.4. If we thus start with an initially empty C_2-cluster, the C_2-population vectors will be restricted to $\mathcal{C}$. One can show that for this model we can even take $D^k \equiv 1$, since due to the definition of coordinate convexity and the definitions of the A^k's the only positive recurrent states $\mathbf{X}$ of this network have a C_2-population vector in $\mathcal{C}$. This model was introduced in LAM [58] though he considered only Poisson arrival processes. He suggested applications in store-and-forward nodes, network flow control and multiprogramming computer systems. Other applications are to be found in interconnected Metropolitan Area Networks (cf. RUBIN AND LEE [78]).

EXAMPLE 3.2.8
Consider the network of Section 3.2.2.1 with two customer classes. Let a, $b \in \mathbb{N}$, $0 < a < b$ and consider the functions

$$D^2\big((m^1, m^2)\big) = 1(m^1 \leq a) + \Big(\frac{b - m^1}{b - a}\Big)1(a < m^1 \leq b), \tag{3.19}$$

$$A^2\big((m^1, m^2)\big) = 1(m^1 \leq b), \tag{3.20}$$

and $D^1 \equiv A^1 \equiv 1$.

In this example class 2 customers are served normally when the number of class 1 customers does not exceed a. If the number of class 1 customers exceeds b, then the servicing of class 2 customers is stopped completely. Between these two levels the service rate for class 2 is gradually decreased. One can view class 1 as having priority over class 2.

EXAMPLE 3.2.9 (ERLANG'S IDEAL GRADING)
Consider a queueing network where C_2 consists of N_2 servers, that each can serve at most one customer at any time. Each time a customer wants to jump from cluster C_1 to C_2, he hunts over R randomly chosen servers for a free server, where $R \leq N_2$ is a fixed integer. If he finds a free server, the customer will receive his service, otherwise he recirculates to C_1. The times spent in clusters C_1 and C_2 are usually referred to as think and busy times. This model can be parametrised by

$$A(\mathbf{m}) = 1 - \binom{\|\mathbf{m}\|}{R} \Big/ \binom{N_2}{R}, \tag{3.21}$$

where $\binom{n}{m} = 0$ for $n < m$ and $\|\mathbf{m}\| = \|(m^1, \ldots, m^K)\| = \sum_{k=1}^{K} m^k$. When C_1 consists solely of N_1 sources with exponentially distributed think times, the model reduces to *Erlang's ideal grading.* To satisfy Condition 3.2.4 it is sufficient to set $D(\mathbf{m}) = 0$ for $\|\mathbf{m}\| > N_2$.

3.2.2.4 The product form equilibrium distribution

In this section we present the main result. We assume without loss of generality that the Markov process corresponding to the queueing network as described in the preceding sections, has a unique equilibrium density function, that is continuously differentiable in all its arguments.

THEOREM 3.2.10 *Assume that we have a queueing network of two clusters of stations, and the departure and arrival blocking functions satisfy Conditions 3.2.4 and 3.2.6. Assume that the service time distributions at FCFS stations are independent of the customer class and exponentially distributed. Let M_0 be the C_2-population vector at time 0, then the equilibrium probability $\pi(\mathbf{X})$ of being in state $\mathbf{X}$, that is characterised by a C_2-population vector $M_2 \in \mathcal{E}$, is*

$$\pi(\mathbf{X}) = G\,F(M_2) \prod_{n=1}^{N} \prod_{s=1}^{k_n} \frac{\theta_n^{x_{ns}}}{f_n(s)} \left(1 - B_n^{x_{ns}}(r_{ns})\right) \tag{3.22}$$

where G is a normalisation constant.

PROOF. See Appendix A.

COROLLARY 3.2.11 *For each state $\mathbf{n}$ the equilibrium distribution is given by*

$$\pi(\mathbf{n}) = G\,F(M_2) \prod_{n=1}^{N} \prod_{s=1}^{k_n} \frac{\theta_n^{x_{ns}}}{\mu_n^{x_{ns}} f_n(s)} \tag{3.23}$$

PROOF. Integrate equation (3.22) from zero to infinity for each r_{ns}, $s = 1, \ldots, k_n$, $n = 1, \ldots, N$. □

Extensions can be given where externally arriving jobs, generated by Poisson sources, can enter a cluster, provided they can depart from the system only from that cluster (cf. Figure 3.5).

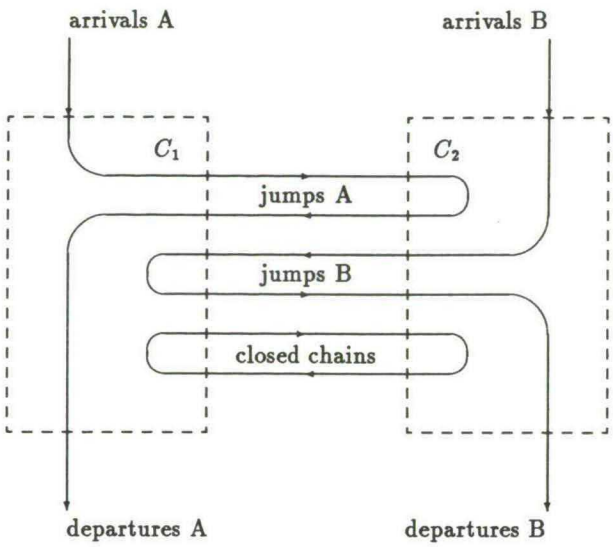

FIGURE 3.5. Mixed open and closed networks.

3.3 STOCHASTIC CONTROL OF QUEUEING SYSTEMS

3.3.1 Introduction

When we have formulated a queueing model for an SPC, we can specify a control algorithm and use methods from performance analysis to compute performance characteristics. With results from Section 3.2 we can evaluate a large class of queueing systems with a control law imposed on the arrival processes. It is not clear beforehand, however, how performance measures like throughput and sojourn times may depend on the parameters of the control law. There exists a method called Perturbation Analysis, that allows estimation of these dependencies from the observations of one simulation of a queueing system. This algorithm can, however, provide only a numerical estimation for a particular realisation and does not give a closed form expression of how performance measures may depend on the parameters of the system (see for example SURI [95] and the references mentioned therein).

If we specify a criterion that we want to be satisfied by the controlled model, then we have to adjust the parameters of the control law and compute the performance measures until we find a suitable setup. For simple queueing systems one can sometimes write down dependencies of the performance measures on the control parameters and use this information to speed up the procedure to search for a control law that satisfies the criterion. Application of this method is unfortunately limited to a small set of queueing systems. It may be clear that if we do not have any knowledge about the dependency of the performance measures on the control parameters, the criterion may be hard or even impossible to achieve.

Consider for instance an M/M/1 queue where the decision to admit or reject customers may be based on the queue length at the moment of an arrival. Since the control law has to specify a decision for each possible value of the queue length, the

number of possible control laws is infinite. Even if we restrict the control laws to be of threshold type — i.e. admit customers only if the momentary queue length does not exceed some threshold value — then there is an infinite set of thresholds and thus of control policies.

An alternative to the performance analysis approach is to use a stochastic control problem formulation. In this approach a (Semi-)Markov Decision Process is constructed by imposing a reward structure on the queueing model. The usual control objective is then formulated as maximising either the expected discounted or average return. The solution of control problems, once a problem formulation is given, proceeds through a number of steps. Here attention is paid to both practical and theoretical considerations of the particular control problem. We can distinguish the following stages for solving the control problem.

(i) *Engineering control synthesis.* In this step a subclass of the class of control laws is selected. This selection is typically based on practical considerations. Engineers who design SPC operating system software are for instance very reluctant to implement a randomised control scheme, even though this scheme may be optimal from a theoretical point of view.

(ii) *Mathematical control synthesis.* Here we try to establish optimality of a particular subclass of the class of control laws. The optimality of a subclass depends of course on the control objective that was specified for the problem. We shall discuss several methodologies for mathematical control synthesis in Section 3.3.4. The results obtained at this stage may be valuable in step (i), since they allow engineers to concentrate on a subclass of control schemes that are known to include the optimal control law.

(iii) *Performance analysis of closed-loop control systems.* At this stage the performance of a (model for a) system is evaluated when a particular control algorithm is implemented. It includes not only the determination of the equilibrium behaviour of the system, but it also focuses on aspects of the dynamic behaviour like stability and robustness. Both mathematical analysis and simulation are tools that can be used at this point of the investigation.

(iv) *Control design.* When we want to implement a control in a real system, we have to determine the numerical parameters of a control law. Several algorithms have been proposed to perform this computation for optimal control problems. We shall discuss a number of algorithms in Section 3.3.3.

(v) *Validation.* At this last stage the control algorithm is tested in practice to see how well it agrees with theoretical results.

It is important in both steps (iv) and (v) that we have access to measurements made of real systems. This may be quite difficult, since either it is (said to be) impossible to perform these measurements or it is not allowed to do this in a situation where the control algorithm is needed most (e.g. when an SPC is overloaded).

The approaches of performance analysis and stochastic control are not entirely distinct nor are the areas in which they can be applied. Performance analysis, despite its limitations, is usually more suited for complex queueing systems. Dependency

of performance measures on the control parameters, however, is often derived with methods from stochastic control, like sample path arguments and forward induction. We present an example of this in Chapter 5. On the other hand there are examples of queueing systems that cannot be addressed by a performance analysis approach, but can be solved by stochastic control methods. Specifically this means that although the optimal control is known exactly, the performance of the optimally controlled system cannot be computed. Examples of this kind are the optimality of the μc rule in priority assignment problems or the join-the-shortest-queue policy in dynamic routing. They shall be discussed in Section 3.3.4.

In this section we give a description of the stochastic control approach to optimal control of queueing systems. The results shall be used in subsequent chapters. We present an introduction to Markov Decision Processes in Section 3.3.2. We shall discuss both the discrete and continuous time versions of the processes. First we formulate the discounted reward control problem and derive the Dynamic Programming equations. The average reward control problem is discussed as a limit of the discounted reward control problem. Methodologies for design and synthesis of optimal control laws are discussed in Sections 3.3.3 and 3.3.4.

3.3.2 Markov Decision Processes

3.3.2.1 Discrete time Markov Decision Processes

In this section we consider a discrete time Markov Decision Process (MDP) that is denoted by $(\mathcal{Y}, \mathcal{S}, \mathcal{A}, p_d, r_d)$. In this notation $\mathcal{Y} = (Y_n, n \in \mathbb{N})$ represents a stochastic process that is observed at times $n \in \mathbb{N}$ to be in a set $\mathcal{S}$ of possible states. The state space $\mathcal{S}$ is assumed to be countable. When the process is in state $s \in \mathcal{S}$ at time n we can choose an action a_n from a set of admissible actions $\mathcal{A}_s \subset \mathcal{A}$, where $\mathcal{A}$ is assumed to be countable and finite. After a decision is taken, the process makes a state transition, the time index increments to $n+1$ and a new action is to be taken.

If the process is in state $s \in \mathcal{S}$ at time n, and action $a \in \mathcal{A}_s$ is taken, then we receive a reward $r_d(s,a)$[1]. We assume that the rewards r_d are positive and bounded, i.e. there exists an $R > 0$ such that $0 \leq r_d(s,a) \leq R$, for all $s \in \mathcal{S}$, $a \in \mathcal{A}_s$. The probability that the process will be in state s' at time $n+1$ is denoted as $p_d(s,a,s')$. Rewards are discounted by a *discount factor* α_d, $0 \leq \alpha_d < 1$, i.e. a reward r received at time n has value $(\alpha_d)^n r$ at time 0.

The actions are determined by a policy. A *policy* or *control law* u is a sequence of decision rules $u_1, u_2, \ldots$, where u_n determines the choice of the action at time n. We allow policies to be randomised, i.e. the decision u_n at time n may be represented as a probability distribution on $\mathcal{A}$. With this in mind one can view a policy as a conditional distribution on the set of actions parametrised by the past history of the process $\mathcal{Y}$. We denote the set of all admissible policies as $\mathcal{U}$. Furthermore a policy is said to be stationary if it is non-randomised and if the action it chooses at any time depends only on the current state of the process. Such a policy may be represented by a mapping $f : \mathcal{S} \to \mathcal{A}$, where $f(s) = a$ means choose action a when the process is in state s. This implies that if a stationary policy f is employed, the process $\mathcal{Y}$ is a homogeneous discrete time Markov process. The set of all stationary policies is

[1] In this notation and others that shall follow the subscripts d and c refer to the discrete time and continuous time Markov Decision Processes, respectively.

denoted as $\mathcal{F}$.

Let $u \in \mathcal{U}$ be some policy and $N \in \mathbf{N}$. The *expected N-horizon* and *infinite horizon* α_d*-discounted return* $V^N_{u,\alpha_d}(s)$ and $V_{u,\alpha_d}(s)$ under policy u when the initial state is $s \in \mathcal{S}$ are defined as

$$V^N_{u,\alpha_d}(s) = E_u\left[\sum_{n=0}^{N-1}(\alpha_d)^n r_d(Y_n, a_n) \,\middle|\, Y_0 = s\right], \qquad s \in \mathcal{S},$$

and

$$V_{u,\alpha_d}(s) = \lim_{N\to\infty} V^N_{u,\alpha_d}(s), \qquad s \in \mathcal{S},$$

where a_n is the decision made at time n and E_u denotes expectation for policy u. The functions V^N_{u,α_d}, $V_{u,\alpha_d} : \mathcal{S} \to \mathbf{R}$ are called the *discounted N-horizon and infinite horizon value functions* for policy u. The parameter N is called the *horizon length.* The value function V_{u,α_d} is well-defined, since the rewards are bounded and $\alpha_d < 1$. The stochastic control problem can now be formulated.

DEFINITION 3.3.1 (DISCOUNTED REWARD CONTROL PROBLEM)
For $0 \le \alpha_d < 1$ *we define the* N-horizon *and* infinite horizon discounted reward control problems $(P^N_{\alpha_d})$ *and* (P_{α_d}) *as*

$(P^N_{\alpha_d})$: *maximise* $V^N_{u,\alpha_d}(s)$ *over* $u \in \mathcal{U}$ *for all* $s \in \mathcal{S}$,

and

(P_{α_d}): *maximise* $V_{u,\alpha_d}(s)$ *over* $u \in \mathcal{U}$ *for all* $s \in \mathcal{S}$,

respectively. The discounted optimal value functions *for the finite and infinite horizon* $V^N_{\alpha_d}$, $V_{\alpha_d} : \mathcal{S} \to \mathbf{R}$ *are defined as*

$$V^N_{\alpha_d}(s) = \sup_{u\in\mathcal{U}} V^N_{u,\alpha_d}(s), \qquad s \in \mathcal{S},$$

and

$$V_{\alpha_d}(s) = \sup_{u\in\mathcal{U}} V_{u,\alpha_d}(s), \qquad s \in \mathcal{S},$$

respectively. Policy u^* *is said to be* optimal *for* $(P^N_{\alpha_d})$ *or* (P_{α_d}) *if*

$$V^N_{\alpha_d}(s) = V^N_{u^*,\alpha_d}(s), \qquad s \in \mathcal{S},$$

or

$$V_{\alpha_d}(s) = V_{u^*,\alpha_d}(s), \qquad s \in \mathcal{S},$$

respectively.

The discounted control problem can be solved by the so-called *Dynamic Programming (DP)* algorithm. This algorithm rests on a very simple idea called Bellman's principle of optimality (see BELLMAN [7]). In words the principle states that the policy u^* that is optimal for the N-horizon control problem should at any time take a decision that is optimal for the next step when one knows that after that step an optimal policy is employed. To demonstrate this principle we need the following preliminaries.

DEFINITION 3.3.2 (SUPREMUM NORM)
On the set of real-valued functions on $\mathcal{S}$ we define the supremum norm as

$$\|g\| = \sup_{s \in \mathcal{S}} |g(s)|, \qquad g : \mathcal{S} \to \mathbf{R}.$$

Let $L(\mathcal{S})$ denote the Banach space of bounded real-valued functions on $\mathcal{S}$, i.e. $g \in L(\mathcal{S})$ if $\|g\| < \infty$.

DEFINITION 3.3.3 *For any stationary policy $f \in \mathcal{F}$, define the operator $T_f : L(\mathcal{S}) \to L(\mathcal{S})$ as*

$$(T_f g)(s) = r_d(s, f(s)) + \alpha_d \sum_{s' \in \mathcal{S}} p_d(s, f(s), s')\, g(s'), \qquad s \in \mathcal{S},\ g \in L(\mathcal{S}).$$

The dynamic programming operator *$T : L(\mathcal{S}) \to L(\mathcal{S})$ is defined as*

$$(Tg)(s) = \max_{a \in \mathcal{A}_s} \left\{ r(s,a) + \alpha_d \sum_{s' \in \mathcal{S}} p_d(s,a,s')\, g(s') \right\}, \qquad s \in \mathcal{S},\ g \in L(\mathcal{S}).$$

Let also $T_f^1 = T_f$, $T^1 = T$, and for $n > 1$ let $T_f^n = T_f(T_f^{n-1})$, $T^n = T(T^{n-1})$.

The following theorem states some properties of the operators. These results are well-known from textbooks on stochastic control (see for instance BERTSEKAS [10, Chapter 5], KUMAR AND VARAIYA [57, Chapter 8] and ROSS [76, Chapter 6]) and they are therefore presented without proof.

THEOREM 3.3.4 *The operators T and T_f, $f \in \mathcal{F}$, are contraction mappings on $L(\mathcal{S})$, i.e. there exist β, β_f, $0 < \beta < 1$, $0 < \beta_f < 1$, such that for all g_1, $g_2 \in L(\mathcal{S})$*

$$\|T_f g_1 - T_f g_2\| \leq \beta_f \|g_1 - g_2\|,$$

$$\|T g_1 - T g_2\| \leq \beta \|g_1 - g_2\|.$$

From the definition of the operator T_f it may be proven that the finite horizon value function V_{f,α_d}^N for a stationary policy $f \in \mathcal{F}$ can be computed recursively by

$$V_{f,\alpha_d}^N(s) = r_d(s, f(s)) + \alpha_d \sum_{s' \in \mathcal{S}} p_d(s, f(s), s') V_{f,\alpha_d}^{N-1}(s'), \qquad N \geq 1,$$

or equivalently

$$V_{f,\alpha_d}^N = T_f V_{f,\alpha_d}^{N-1}, \qquad N \geq 1,$$

where $V_{f,\alpha_d}^0 \equiv 0$. The DP principle that was mentioned before, says that the optimal value function for a finite horizon control problem can be computed recursively as

$$V_{\alpha_d}^N(s) = \max_{a \in \mathcal{A}_s} \left\{ r_d(s,a) + \alpha_d \sum_{s' \in \mathcal{S}} p_d(s,a,s') V_{\alpha_d}^{N-1}(s') \right\}, \qquad N \geq 1,$$

or equivalently

$$V_{\alpha_d}^N = T\, V_{\alpha_d}^{N-1}, \qquad N \geq 1,$$

again with $V_{\alpha_d}^0 \equiv 0$. If we let $N \to \infty$, then we get the following functional equations.

THEOREM 3.3.5 *If the lump rewards $r_d(s,a)$ are bounded in $s \in \mathcal{S}$ and $a \in \mathcal{A}_s$, then the infinite horizon discounted value function for a stationary policy $f \in \mathcal{F}$ is the unique solution V_{f,α_d} in $L(\mathcal{S})$ of the set of equations*

$$V_{f,\alpha_d}(s) = r_d(s, f(s)) + \alpha_d \sum_{s' \in \mathcal{S}} p_d(s, f(s), s') V_{f,\alpha_d}(s'), \qquad s \in \mathcal{S}, \tag{3.24}$$

or equivalently $V_{f,\alpha_d} = T_f V_{f,\alpha_d}$.

PROOF. See BERTSEKAS [10] and ROSS [76, 77].

An analogous set of equations can be derived for the infinite horizon optimal value function. Let $J \in L(\mathcal{S})$ and consider the set of equations

$$(TJ)(s) = \max_{a \in \mathcal{A}_s} \left\{ r_d(s,a) + \alpha_d \sum_{s' \in \mathcal{S}} p_d(s,a,s') J(s') \right\}, \qquad s \in \mathcal{S}. \tag{3.25}$$

We can interpret these equations as follows. For a given $s \in \mathcal{S}$, the action $a \in \mathcal{A}_s$ that maximises the right-hand side of (3.25) is optimal for the control problem with horizon length 1, initial state s, lump reward r_d, where we receive an additional terminal reward $\alpha_d J$. Analogously to Definition 3.3.3 we have $T^N J(s)$ the optimal expected return for an N-horizon control problem with initial state s, lump reward r_d and a terminal reward $(\alpha_d)^N J$. The ordinary N-horizon control problem is in fact a special case with $J \equiv 0$. The following theorem shows that $T^N J$ converges to the optimal value function V_{α_d} for any choice of $J \in L(\mathcal{S})$.

THEOREM 3.3.6 (DYNAMIC PROGRAMMING EQUATIONS (DPE))

(i) For any $J \in L(\mathcal{S})$, $V_{\alpha_d} = \lim_{N \to \infty} T^N J$.

(ii) V_{α_d} satisfies the so-called discrete time dynamic programming equation

$$V_{\alpha_d}(s) = \max_{a \in \mathcal{A}_s} \left\{ r_d(s,a) + \alpha_d \sum_{s' \in \mathcal{S}} p_d(s,a,s') V_{\alpha_d}(s') \right\} \tag{3.26}$$

or equivalently $V_{\alpha_d} = T V_{\alpha_d}$.

(iii) A stationary policy $f \in \mathcal{F}$ is α_d-optimal if and only if in each state $s \in \mathcal{S}$ it chooses the action that maximises the right-hand side of (3.26).

PROOF. See BERTSEKAS [10] and ROSS [76, 77].

Although the discounted control problem is very appealing from a theoretical point of view, in practice we are often more interested in maximising a non-discounted return. For $u \in \mathcal{U}$ and $N \in \mathbb{N}$ define $W_u^N(s)$, the *total expected return over horizon length N* with initial state $s \in \mathcal{S}$, as

$$W_u^N(s) = E_u \left[\sum_{n=0}^{N} r_d(Y_n, a_n) \,\middle|\, Y_0 = s \right], \qquad s \in \mathcal{S}.$$

The average return $W_u(s)$ for policy $u \in \mathcal{U}$ with initial state $s \in \mathcal{S}$ is defined as

$$W_u(s) = \liminf_{N\to\infty} \frac{1}{N+1} W_u^N(s), \qquad s \in \mathcal{S}.$$

DEFINITION 3.3.7 (AVERAGE REWARD CONTROL PROBLEM)
The average reward control problem (P) *is defined as*

(P): *maximise* $W_u(s)$ *over* $u \in \mathcal{U}$ *for all* $s \in \mathcal{S}$.

The optimal average return $W : \mathcal{S} \to \mathbf{R}$ *is defined as*

$$W(s) = \sup_{u\in\mathcal{U}} W_u(s), \qquad s \in \mathcal{S}. \tag{3.27}$$

A policy $u^* \in \mathcal{U}$ *is said to be* average optimal *if*

$$W(s) = W_{u^*}(s), \qquad s \in \mathcal{S}. \tag{3.28}$$

Although it is possible to write down a set of functional equations for the average optimal value functions, the average control problem is usually treated as the limit of a discounted control problem. The following condition is sufficient to make the discounted optimal policy converge to the average optimal policy (cf. ROSS [76, Theorem 6.18]).

THEOREM 3.3.8 (ROSS' CONDITION)
Let the rewards $r_d(s,a)$ *be bounded in* $s \in \mathcal{S}$ *and* $a \in \mathcal{A}_s$. *If there exists a reference state* $s_0 \in \mathcal{S}$ *and a constant* $M < \infty$ *such that*

$$|V_{\alpha_d}(s) - V_{\alpha_d}(s_0)| < M,$$

for all $\alpha_d > 0$ *and* $s \in \mathcal{S}$, *then the optimal average return* $W(s)$ *is independent of the initial state* s *and* $W(s) \equiv V \equiv \lim_{\alpha_d\uparrow 1}(1-\alpha_d)V_{\alpha_d}(s_0)$. *Moreover, the* α_d*-discounted optimal policy converges to the average optimal policy as* $\alpha_d \uparrow 1$.

Unfortunately this condition is seldom satisfied in stochastic control problems for queueing systems. Alternative conditions for convergence have been derived (see for instance BORKAR [15] and HAJEK [40]). A more appropriate set of conditions for control problems in queueing systems is due to SENNOTT [85, 86, 87].

THEOREM 3.3.9 (SENNOTT'S CONDITIONS)
If the rewards $r_d(s,a)$ *are bounded in* $s \in \mathcal{S}$ *and* $a \in \mathcal{A}_s$ *and there exists a reference state* $s_0 \in \mathcal{S}$ *and finite non-negative constants* B, C_s, D_s, $s \in \mathcal{S}$ *such that*

$$V_{\alpha_d}(s) - V_{\alpha_d}(s_0) \le B, \qquad s \in \mathcal{S},\ \alpha_d > 0, \tag{3.29a}$$

$$V_{\alpha_d}(s) - V_{\alpha_d}(s_0) \ge -C_s, \qquad s \in \mathcal{S},\ \alpha_d > 0, \tag{3.29b}$$

$$\sum_{s'\in\mathcal{S}} p_d(s,a,s')C_{s'} \le D_s, \qquad s \in \mathcal{S},\ a \in \mathcal{A}_s, \tag{3.29c}$$

then the average optimal value function $W(s)$ *is independent of the initial state* s *and* $W(s) \equiv W \equiv \lim_{\alpha_d\uparrow 1}(1-\alpha_d)V_{\alpha_d}(s_0)$. *Under these conditions the* α_d*-discounted optimal policy converges to the average optimal policy.*

3.3.2.2 Continuous time Markov Decision Processes

In this section we consider a continuous time Markov Decision Process (MDP) that is denoted by $(\mathcal{X}, \mathcal{S}, \mathcal{A}, \gamma, p_c, r_c)$. In this notation $\mathcal{X} = (X_t, t \geq 0)$ represents a continuous time stochastic process that is observed at times $t \geq 0$ to be in a set $\mathcal{S}$ of possible states. Here $\mathcal{S}$ is the same as the state space of the discrete time process. When in state $s \in \mathcal{S}$ we can choose an action from a set of admissible actions $\mathcal{A}_s \subset \mathcal{A}$, where $\mathcal{A}$ is again assumed countable and finite. At some time after a decision is taken the process will move to another state and a new action is to be taken.

If the process $\mathcal{X}$ is in state $s \in \mathcal{S}$ and action $a \in \mathcal{A}_s$ is taken, then we receive a lump reward $r_c(s,a)$. After a random time that is exponentially distributed with rate $\gamma(s,a)$ the process will move to a new state $s' \in \mathcal{S}$ with probability $p_c(s,a,s')$. Without loss of generality we assume that $p_c(s,a,s) = 0$. We also assume that there exist finite bounds Γ, $R > 0$ such that $\gamma(s,a) < \Gamma$ and $0 < r(s,a) < R$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}_s$. This guarantees that in a finite amount of time only a finite number of transitions can occur and only a finite return is gained.

For the continuous time MDP a policy is defined analogously to the discrete time process. Denote by $t_1, t_2, \ldots$, the instants at which the process $\mathcal{X}$ makes a transition. A *policy* u is a sequence of decision rules $u_0, u_1, u_2, \ldots$, where u_n determines the choice of the action immediately after transition time t_n. Action u_0 is the decision at time 0. In this sense a policy is a conditional distribution on the set of actions parametrised by the past history of the process $\mathcal{X}$. A stationary policy is again a policy that uses as information only the current state of the process. Again $\mathcal{U}$ and $\mathcal{F}$ are used to denote the set of all admissible and stationary policies, respectively.

Let $u \in \mathcal{U}$ be some policy. We define the *total expected return* $W_u^T(s)$ up to time $T > 0$ for policy u when the initial state is $s \in \mathcal{S}$ as

$$W_u^T(s) = E_u\left[\sum_{\{n \mid t_n \leq T\}} r_c(X_{t_n-}, a_{t_n-}) \,\middle|\, X_0 = s \right], \qquad s \in \mathcal{S}, T \geq 0. \tag{3.30}$$

Let $\alpha_c > 0$ be the *discount rate*, i.e. a reward r received at time t has value $e^{-\alpha_c t} r$ at time 0. The *total α_c-discounted returns* V_{u,α_c}^T, $V_{u,\alpha_c} : \mathcal{S} \to \mathbf{R}$ under policy u for horizon length $T > 0$ and the infinite horizon are defined as

$$V_{u,\alpha_c}^T(s) = \int_0^T e^{-\alpha_c t} \, d_t W_u^t(s), \qquad s \in \mathcal{S}. \tag{3.31}$$

and

$$V_{u,\alpha_c}(s) = \int_0^\infty e^{-\alpha_c t} \, d_t W_u^t(s), \qquad s \in \mathcal{S}. \tag{3.32}$$

Note that (3.32) is well defined, since $\alpha_c > 0$ and rewards r_c are bounded by R. The function $V_{u,\alpha_c} : \mathcal{S} \to \mathbf{R}$ is usually called the *continuous time discounted value function* for policy u.

DEFINITION 3.3.10 (DISCOUNTED REWARD CONTROL PROBLEM)
For $\alpha_c > 0$ the finite *and* infinite horizon discounted reward control problems $(P_{\alpha_c}^T)$ *and* (P_{α_c}) *are defined as*

$(P_{\alpha_c}^T)$: *maximise* $V_{u,\alpha_c}^T(s)$ *over* $u \in \mathcal{U}$ *for all* $s \in \mathcal{S}$.

and

$$(P_{\alpha_c})\text{: maximise } V_{u,\alpha_d}(s) \text{ over } u \in \mathcal{U} \text{ for all } s \in \mathcal{S}.$$

The discounted optimal value functions $V^T_{\alpha_c}$, $V_{\alpha_c} : \mathcal{S} \to \mathbf{R}$ *for horizon length* $T > 0$ *and the infinite horizon are defined as*

$$V^T_{\alpha_c}(s) = \sup_{u \in \mathcal{U}} V^T_{u,\alpha_c}(s), \qquad s \in \mathcal{S}, \tag{3.33}$$

and

$$V_{\alpha_c}(s) = \sup_{u \in \mathcal{U}} V_{u,\alpha_c}(s), \qquad s \in \mathcal{S}, \tag{3.34}$$

respectively. Policy u^* *is called optimal for problem* $(P^T_{\alpha_c})$ *or* (P_{α_c}) *if*

$$V^T_{\alpha_c}(s) = V^T_{u^*,\alpha_c}(s), \qquad s \in \mathcal{S}, \tag{3.35}$$

or

$$V_{\alpha_c}(s) = V_{u^*,\alpha_c}(s), \qquad s \in \mathcal{S}, \tag{3.36}$$

respectively.

For practical reasons it is usually more convenient to transform the continuous time process $\mathcal{X}$ into an equivalent process that operates with a discrete time axis. This can be accomplished by *uniformisation* (see for instance LIPPMAN [61] and SERFOZO [88]).

THEOREM 3.3.11 (UNIFORMISATION)
Let $\mathbf{X} := (\mathcal{X}, \mathcal{S}, \mathcal{A}, \gamma, p_c, r_c)$ *be a continuous time MDP, where transition rates are bounded by* Γ *and lump rewards are bounded by* R. *Consider a discrete time MDP* $\mathbf{Y} := (\mathcal{Y}, \mathcal{S}, \mathcal{A}, p_d, r_d)$ *constructed as follows. The transition probabilities* $p_d(s,a,s')$ *are defined as*

$$p_d(s,a,s') = \begin{cases} \dfrac{\Gamma - \gamma(s,a)}{\Gamma}, & \text{if } s = s', \\ \dfrac{\gamma(s,a)\, p_c(s,a,s')}{\Gamma}, & \text{if } s \neq s', \end{cases} \tag{3.37}$$

and the lump rewards r_d *are defined as*

$$r_d(s,a) = r_c(s,a)/(\Gamma + \alpha_c). \tag{3.38}$$

If the discount rate for $\mathbf{X}$ *is* α_c, *then define the discount factor* α_d *for* $\mathbf{Y}$ *as*

$$\alpha_d = \Gamma/(\Gamma + \alpha_c). \tag{3.39}$$

If in both processes $\mathbf{X}$ *and* $\mathbf{Y}$ *actions are chosen according to a stationary policy* f, *then the discounted value functions* V_{f,α_d} *and* V_{f,α_c} *are equal for both Markov Decision Processes.*

PROOF. See LIPPMAN [61] and SERFOZO [88].

Note that the multiplication of $r_c(s,a)$ in (3.38) with $(\Gamma+\alpha_c)^{-1}$ can be relaxed, since this factor is constant for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ and hence does not affect the optimality of policies. In fact, omitting the multiplication yields for all policies $u \in \mathcal{U}$, $V_{u,\alpha_d} = (\Gamma+\alpha_c)V_{u,\alpha_c}$.

With Theorem 3.3.11 we can immediately state the continuous time equivalents of Theorems 3.3.5 and 3.3.6.

THEOREM 3.3.12 *If the lump rewards r_d are bounded in $s \in \mathcal{S}$ and $a \in \mathcal{A}_s$, then the infinite horizon value function for a stationary policy $f \in \mathcal{F}$ is the unique solution V_{f,α_c} in $L(\mathcal{S})$ of*

$$\alpha_c V_{f,\alpha_c}(s) = r_c(s,f(s)) + \gamma(s,f(s)) \sum_{s' \neq s} p_c(s,f(s),s')\big[V_{f,\alpha_c}(s') - V_{f,\alpha_c}(s)\big]$$

PROOF. Since by Theorem 3.3.11 V_{f,α_c} and V_{f,α_d} are equal, we have

$$\begin{aligned} & V_{f,\alpha_c}(s) \\ &= r_d(s,f(s)) + \alpha_d \sum_{s' \neq s} p_d(s,f(s),s') V_{f,\alpha_c}(s') + \alpha_d\, p_d(s,f(s),s) V_{f,\alpha_c}(s) \\ &= \frac{r_c(s,f(s))}{\Gamma+\alpha_c} + \sum_{s \neq s'} \frac{\gamma(s,f(s)) p_c(s,f(s),s')}{\Gamma+\alpha_c} V_{f,\alpha_c}(s') + \frac{\Gamma - \gamma(s,f(s))}{\Gamma+\alpha_c} V_{f,\alpha_c}(s). \quad \square \end{aligned}$$

THEOREM 3.3.13 (DYNAMIC PROGRAMMING EQUATIONS (DPE))
If the lump rewards $r_c(s,a)$ are bounded in $s \in \mathcal{S}$ and $a \in \mathcal{A}_s$, then the optimal value function V_{α_c} is equal to V_{α_d} and it is the unique bounded solution to the continuous time dynamic programming equations

$$\alpha_c V_{\alpha_c}(s) = \max_{a \in \mathcal{A}_s} \left\{ r_c(s,a) + \gamma(s,a) \sum_{s' \neq s} p_c(s,a,s')\big[V_{\alpha_c}(s') - V_{\alpha_c}(s)\big] \right\}. \tag{3.40}$$

PROOF. With the uniformisation theorem the proof proceeds analogously to the proof of Theorem 3.3.12. $\square$

Analogously to the discrete time MDP we can define an average reward control problem for continuous time MDP's. Recall the definition of the cumulative reward W_u^T in (3.30). For a policy $u \in \mathcal{U}$ we define the *average return* $W_u(s)$ when the initial state is $s \in \mathcal{S}$ and policy u is employed as

$$W_u(s) = \liminf_{T \to \infty} \frac{W_u^T(s)}{T}, \qquad s \in \mathcal{S}. \tag{3.41}$$

DEFINITION 3.3.14 (AVERAGE REWARD CONTROL PROBLEM)
The average reward control problem (P) *is defined as*

(P): *maximise $W_u(s)$ over $u \in \mathcal{U}$ for all $s \in \mathcal{S}$.*

The average optimal value function $W : \mathcal{S} \to \mathbf{R}$ *is defined as*

$$W(s) = \sup_{u \in \mathcal{U}} W_u(s), \qquad s \in \mathcal{S}. \tag{3.42}$$

A policy $u^* \in \mathcal{U}$ is called optimal for (P) if

$$W(s) = W_{u^*}(s), \qquad s \in \mathcal{S}. \tag{3.43}$$

The convergence results of Theorems 3.3.8 and 3.3.9 carry over to the continuous time MDP provided we take the limit as $\alpha_c \downarrow 0$ as the analogue of $\alpha_d \uparrow 1$.

We conclude this section with some remarks on our approach to the modelling of MDP's.

REMARK 3.3.15 In the continuous time version of the MDP we can allow the process to jump without a change of state. Assume that we have a continuous time stochastic process $\mathcal{X}' = (X_t, t \geq 0)$ that allows $p_c'(s,a,s) > 0$. One can easily show that this process is stochastically equivalent to the process $\mathcal{X}$ with transition rate $\gamma_c(s,a)$ out of state $s \in \mathcal{S}$ equal to

$$\gamma_c(s,a) = \gamma_c'(s,a)\big(1 - p_c'(s,a,s)\big)$$

and transition probabilities $p_c(s,a,s')$ given by

$$p_c(s,a,s') = \begin{cases} 0, & \text{if } s = s', \\ \dfrac{p_c'(s,a,s')}{1 - p_c'(s,a,s)}, & \text{if } s \neq s'. \end{cases}$$

REMARK 3.3.16 The reward function $r_c(s,a)$ can model a more complex reward structure than merely a lump reward. Assume that we have a continuous time process $\mathcal{X} = (X_t, t \geq 0)$. When the process is in state $s \in \mathcal{S}$, we choose action $a \in \mathcal{A}_s$ and if the next state is $s' \in \mathcal{S}$, then we receive

- A lump reward $r_1(s,a,s')$ at the moment of the decision.
- A reward $r_2(s,a,s')$ per time unit until the transition to s'.
- A lump reward $r_3(s,a,s')$ at the moment of the transition.

The reward structure can be represented in our model by choosing

$$r(s,a) = \sum_{s' \in \mathcal{S}} \left\{ r_1(s,a,s') + \frac{r_2(s,a,s')}{\gamma(s,a) + \alpha_c} + \frac{\gamma(s,a) r_3(s,a,s')}{\gamma(s,a) + \alpha_c} \right\} p_c(s,a,s').$$

REMARK 3.3.17 The combination of a transition rate $\gamma(s,a)$ and transition probabilities $p_c(s,a,s')$ allows an alternative formulation for the transition mechanism. Assume that we have a continuous time process $\mathcal{X}' = (\mathcal{X}_t', t \geq 0)$ where, if we take action $a \in \mathcal{A}_s$ in state $s \in \mathcal{S}$, the process makes a transition to state s' after an exponentially distributed time with rate $\gamma'(s,a,s')$. Assume that only a finite number of states $s' \in \mathcal{S}$ have $\gamma(s,a,s') > 0$. Since the minimum of a finite number of exponentially distributed random variables is again exponentially distributed, we can transform this process into our model by choosing

$$\gamma(s,a) = \sum_{s' \in \mathcal{S}} \gamma'(s,a,s')$$

and

$$p_c(s,a,s') = \frac{\gamma'(s,a,s')}{\gamma(s,a)}.$$

Note that the DPE equations in this setting are

$$\alpha_c V_{\alpha_c}(s) = \max_{a \in \mathcal{A}_s} \left\{ \sum_{s' \in \mathcal{S}} \gamma'(s,a,s') \left[V_{\alpha_c}(s') - V_{\alpha_c}(s) \right] \right\},$$

where we can even allow dummy transitions, i.e $\gamma'(s,a,s) > 0$.

REMARK 3.3.18 In a strict mathematical sense the MDP formulation is not completely adequate. We have to show for instance that all the random variables are well-defined with a well-defined finite expected value. In the framework of probability theory we would have to introduce a proper probability space and put measurability conditions on policies and rewards. We will, however, not attempt to present a rigorous formulation of stochastic control problems for general state and action spaces, since these difficulties are mainly technical and do not affect the results we present in the remainder of this thesis. For our purposes it suffices to state that under the boundedness of rewards and transition rates, plus the countability of the state and action spaces, all the mathematical difficulties disappear. The reader who wants to enjoy a thorough mathematical treatment of this subject is referred to BERTSEKAS AND SHREVE [11] and STRIEBEL [94].

3.3.3 Design of optimal policies

The functional equations of Theorems 3.3.6 and 3.3.13 provide necessary and sufficient conditions for the optimal value functions and the optimal policies. The maximisation operator in the definition of T makes the DPE nonlinear, and because of this, the equations are intrinsically difficult to solve. The computation of the numerical parameters of the optimal control, usually called the *design problem* of stochastic control, is therefore a non trivial task.

In a number of applications we are, however, not as much interested in the numerical parameters of the optimal policy as in structural properties of the optimal policy. This is called the *synthesis problem* of stochastic control. Knowledge of such structures may be valuable to improve the speed of standard methods for the design problem.

In this section we discuss the design problem of stochastic control. The synthesis problem shall be addressed in Section 3.3.4. Without loss of generality we discuss these problems only for the discrete time MDP and drop the subscript d from all notation.

3.3.3.1 Successive approximation

The successive approximation method is essentially the DP algorithm applied over a finite horizon, that yields in the limit, as we let the horizon length go to infinity, the optimal value function and policy of the infinite horizon control problem.

As we have seen in Theorem 3.3.6 the sequence $T^n J$ in $L(\mathcal{S})$ converges to the optimal value function V_α for any choice of $J \in L(\mathcal{S})$. This is the foundation of the successive approximation method.

ALGORITHM 3.3.19 (SUCCESSIVE APPROXIMATION)

(i) Initialisation step: Let $n = 0$ and take an arbitrary $V^0 \in L(\mathcal{S})$.

(ii) Value iteration step: Increase n by 1 and compute $V^n = TV^{n-1}$ from

$$(TV^{n-1})(s) = \max_{a \in \mathcal{A}_s} \left\{ r(s,a) + \alpha \sum_{s' \in \mathcal{S}} p(s,a,s') V^{n-1}(s') \right\}, s \in \mathcal{S},$$

until convergence.

The successive approximation method is sometimes also called the value iteration algorithm.

One serious drawback of the successive approximation method is the choice of the convergence criterion. As the procedure yields the optimal value function V only in the limit — unless we are lucky enough to choose $V^0 \equiv V$ — we have to settle for an approximation of V if we want to run the algorithm only for a finite time. On the other hand the algorithm provides the optimal policy within a finite number of iterations, provided the state space is finite (cf. BERTSEKAS [10, Chapter 5, Problem 14]).

The contracting property of the DP operator T, however, can provide us with upper and lower bounds for V_α as the algorithm proceeds. If we denote c_-^n, c_+^n, $n \in \mathbf{N}$ as

$$\begin{aligned} c_-^n &= \frac{\alpha}{1-\alpha} \inf_{s \in \mathcal{S}} \left\{ V^n(s) - V^{n-1}(s) \right\}, \\ c_+^n &= \frac{\alpha}{1-\alpha} \sup_{s \in \mathcal{S}} \left\{ V^n(s) - V^{n-1}(s) \right\}, \end{aligned} \tag{3.44}$$

one can show that

$$\begin{aligned} V^n(s) + c_-^n \;&\leq\; V^{n+1}(s) + c_-^{n+1} \;\leq\; V(s) \\ &\leq\; V_{n+1}(s) + c_+^{n+1} \;\leq\; V^n(s) + c_+^n, \end{aligned} \tag{3.45}$$

for all $s \in \mathcal{S}$. Notice that the error bounds in (3.44) can be computed as a byproduct of the computations in the value iteration step of Algorithm 3.3.19. The bounds can also be useful in the design of a convergence criterion. If for instance we want the maximally allowed error to be smaller than ε, then a sufficient convergence criterion is to stop when $c_+^n - c_-^n \leq \varepsilon$.

The main advantage of the algorithm is the simplicity of the computations involved in step (ii) (from a numerical point of view that is). The procedure can therefore be implemented without numerical instability for MDP's with large state and action spaces. Moreover, the algorithm can be implemented quite easily on a parallel computer.

Note that, although the procedure is well-defined for an arbitrarily large state space $\mathcal{S}$, implementation requires $\mathcal{S}$ either to be finite or to be truncated to a finite space.

3.3.3.2 Policy iteration

The policy iteration method is an alternative approach for the computation of the optimal value function and optimal policy for a discounted infinite horizon control problem. The method is iterative in nature and determines in each step a new policy that has a higher value function than the previously computed ones. It is defined as follows.

ALGORITHM 3.3.20 (POLICY ITERATION)

(i) Initialisation step: Let $n = 0$ and take an arbitrary stationary policy $f_0 \in \mathcal{F}$.

(ii) Policy evaluation step: Compute the value function V_{f_n} from the equation $V_{f_n} = TV_{f_n}$.

(iii) Policy improvement step: Increase n by 1 and determine a stationary policy f_n that satisfies $T_{f_n}V_{f_{n-1}} = TV_{f_{n-1}}$. If f_{n-1} is a possible candidate, then choose $f_n = f_{n-1}$.

(iv) If $f_n = f_{n-1}$ then stop, else continue with step (ii).

Each time we perform step (iii) of the algorithm, we construct a new policy that, when it is not equal to the previous one, has a larger value function. This is proven in BERTSEKAS [10, Section 5.2, Proposition 7].

The policy iteration method is useful only for MDP's with a finite state and action space. In that case the number of stationary policies is finite and the algorithm is guaranteed to finish within a finite number of iterations. This property is its main advantage over the successive approximation method. Its main drawback is somewhat hidden in step (ii), where a set of linear equations has to be solved for the value function V_{f_n}. The dimension of this system is equal to the number of states and thus the method becomes numerically unattractive when the state space is large.

3.3.3.3 Modified algorithms

To overcome the relatively slow convergence of the successive approximation method and the numerical instability of the policy iteration algorithm, some modifications to these procedures have been proposed that try to combine the assets of both algorithms. These methods are referred to as both modified value and policy iteration in literature (cf. BERTSEKAS[10, Section 5.2], PUTERMAN AND SHIN [70] and TIJMS [97, Section 3.4]).

Assume that we have a finite state space in which the states are numbered as $\mathcal{S} = \{1, \ldots, m\}$. The family of modified algorithms can be defined as follows.

ALGORITHM 3.3.21 (MODIFIED VALUE/POLICY ITERATION)
Let $\nu : \mathbb{N}_m \to \mathbb{N} \cup \{0\}$ and $M \in \mathbb{N}$.

(i) Initialisation step: Let $V^0 \equiv 0$, $n = 0$.

(ii) Policy improvement step: Increase n by 1 and determine a stationary policy f_n that satisfies $T_{f_n}V^{n-1} = TV^{n-1}$.

(iii) Modified value iteration step: Compute $V^n = \mathcal{T}_{f_n}^M V^{n-1}$, implicitly defined by

$$(\mathcal{T}_f V)(i) = \tag{3.46}$$

$$r(i, f(i)) + \alpha\left\{ \sum_{j=1}^{\nu(i)} p(i, f(i), j)(\mathcal{T}_f V)(j) + \sum_{j=\nu(i)+1}^{m} p(i, f(i), j)V(j) \right\}$$

for $i = 1, \ldots, m$, until convergence.

If we take $\nu \equiv 0$, then $\mathcal{T}_f = T_f$, $f \in \mathcal{F}$. For $M = 1$ the algorithm reduces to ordinary successive approximation. For this case step (iii) of Algorithm 3.3.21 is called the *Jacobi form of successive approximation.* For $M > 1$ the algorithm can be regarded as a modification of the policy iteration algorithm, where the computation of V_{f_n} in the policy evaluation step is approximated by M steps of Jacobi iterations. The bounds in (3.44) can be used to determine an appropriate value of M dynamically.

If we take $\nu(i) = i - 1$, then in the modified value iteration step V^n is computed as

$$V^n(i) = r(i, f(i)) + \alpha\left\{ \sum_{j=1}^{i-1} p(i, f(i), j)V^n(j) + \sum_{j=i}^{m} p(i, f(i), j)V^{n-1}(j) \right\},$$

for $i = 1, \ldots, m$. This modification is called the *Gauss-Seidel version of successive approximation.* The key idea is to compute the value iteration step for one state at a time while incorporating the interim results into the computation. The computation is as easy as the value iteration step in ordinary successive approximation. In BERTSEKAS [10, Section 5.2, Prop. 5 and 6] it is shown that the V^n converge to the optimal value function V and that the convergence is faster than for the unmodified value iteration algorithm.

If we take $\nu(i) = i$, then we get as step (iii) the computation of

$$V^n(i) = r(i, f(i)) + \alpha\left\{ \sum_{j=1}^{i} p(i, f(i), j)V^n(j) + \sum_{j=i+1}^{m} p(i, f(i), j)V^{n-1}(j) \right\}$$

for $i = 1, \ldots, m$. This is equivalent to

$$V^n(i) = \frac{r(i, f(i)) + \alpha\left\{ \sum_{j=1}^{i-1} p(i, f(i), j)V^n(j) + \sum_{j=i+1}^{m} p(i, f(i), j)V^{n-1}(j) \right\}}{1 - \alpha\, p(i, f(i), i)}$$

for $i = 1, \ldots, m$. When $M = 1$ this is called the overrelaxation form of the successive approximation method. A variation on this procedure is presented in TIJMS [97, Section 3.4, p. 195]. For $M > 1$ this modification can be seen as policy iteration where the policy evaluation step is approximated by M steps of successive overrelaxation.

3.3.4 Synthesis of optimal policies

A numerical solution of a stochastic control problem may be very useful in a concrete application, but it may not give much insight into how a queueing system should be controlled. Such insight may be gained by the solution of the *synthesis problem* of stochastic control. In the synthesis problem structural properties of optimal policies are derived. As examples of such properties one can think of optimality of admission

policies that are determined by thresholds or coordinate convex access regions, or assignment policies determined by switching curves.

Consider as an example the M/M/1 queue as mentioned before where new arrivals can be rejected or admitted and this decision is based on the momentary queue length at the moment of an arrival. A stationary policy f can be represented as a function from $\mathbb{N}$ to $\{0,1\}$, where $f(i) = 0$ or 1 corresponds to rejecting or admitting a customer when the queue length is i, respectively. For any stationary policy f we can compute the value function V_f and check whether the policy is optimal. The policy f is optimal if and only if the action that maximises the right-hand side of (3.26) is the same as the action prescribed by policy f. If for at least one state the actions are different, f is not the optimal policy and we have to make another guess for f. If we are able to show that the optimal policy is in a strict subset of the set of admissible (stationary) policies, then we can use this extra information to speed up the search for the optimal policy. In the example of the M/M/1 queue, we can think of the optimal policy lying in the class of *threshold policies*, i.e. policies f of the form $f(i) = 1(i < L)$ for some $L \in \mathbb{N}$. Clearly the set of threshold policies is a strict subset of the set of admissible policies. In Chapter 4 we present an example of a control problem for an M/M/1 queue where the optimal policy is a threshold policy. Moreover, we present an algorithm to search for the optimal threshold that explicitly uses this information.

The derivation of structural properties for optimal control problems in queueing systems has been limited to systems with small dimensions of the state space. Although some results are known on problems with a multi-dimensional state space, a general approach for the generalisation of existing results to higher dimensions is still lacking.

In this section we shall discuss some methodologies for the synthesis problem of optimal control, namely backward recursion in Section 3.3.4.1, interchange arguments in Section 3.3.4.2 and forward induction in Section 3.3.4.3.

3.3.4.1 Backward recursion

The *backward recursion* approach to proving structural properties of optimal policies is based on the successive approximation method as described in Section 3.3.3.1. Since the method is based on the fundamental optimality principle that underlies the DP equations, it is also called the DPE methodology.

With the successive approximation method the optimal value function for the infinite horizon was computed as the limit of the finite horizon optimal value function. These finite horizon optimal value functions were computed recursively by induction on the horizon length. The key idea of the backward recursion method is to prove that some property is satisfied by the finite horizon optimal policy for any horizon length. Usually one can transform properties of optimal policies into properties of the optimal value function, and try to prove these by induction on the horizon length. The properties that are mostly searched for often reflect that the value function has to satisfy some form of monotonicity, like concavity, unimodality or submodularity. These properties can be translated into "monotonicity" of the optimal policies, like optimality of a threshold policy or the existence of a switching curve.

Since the proofs are usually by induction, one has to choose a suitable starting function V^0 in step (i) of the algorithm and show that the DP operator preserves the desired property. In most cases the monotonicity properties are satisfied by the

all-zero function and $V^0 \equiv 0$ is an appropriate choice.

We shall now discuss several applications of the backward recursion method. It is used in STIDHAM AND PRABHU [92] to prove concavity of the N-stage optimal value function in a queueing system where the state is the work in the system and the actions are to decide what fraction of an arriving customer's workload should be admitted.

Another example of an application is the proof of the convexity of the N-stage optimal value function in a batch GI/M/1 queue where part of the batch may be admitted and a reward is gained that is proportional to the number of customers admitted (cf. STIDHAM [90]). As a consequence the optimal policy is monotonic in the sense that as the number of customers in the queue increases new customers are less likely to be admitted.

In JOHANSEN AND STIDHAM [47] backward recursion is applied to a stochastic system with a very general input and output structure. In the model a random reward is given for an admitted customer and a waiting cost is incurred that may depend on the state variable. Concavity of the objective function is used to prove the optimality of threshold policies for e.g. a GI/G/1 queue and models with compound input and output processes.

In HAJEK [40] a queueing system is considered that consists of two single server queues, each with their own Poisson arrival process. Arrivals from a third stream can be routed to one of the two queues. Also a third server is available to be assigned to one of the two queues. The costs incurred are linear in the queue lengths and it is shown that the optimal value function is convex. As a result the optimal policy is described by a switching curve, i.e. a curve in $\mathbb{N}^2$ that prescribes to which queue the third server or an arriving customer should be assigned.

In LIN AND KUMAR [60] the method is used to derive the structure of the optimal policy in a queueing system with two heterogeneous servers that can serve one queue of customers and where the objective is to minimise the mean sojourn times of customers. The optimal policy has a threshold structure, namely the slow server should be activated only when the queue length exceeds some threshold value.

The recursion procedure is used in BARAS *et al.* [2] to prove the optimality of the μc-rule in a system with two competing queues. This problem that was introduced in KLIMOV [55], is a dynamic priority assignment problem with different classes of customers. There is one server attending the customers and service times are exponentially distributed with class-dependent rates μ_i for class i. A waiting cost c_i is incurred per time unit for each customer of class i. The results in [2] state that with two customer classes the discounted costs over the finite and infinite horizon are minimised by applying the μc-rule: among the customers in the queue always serve the ones from the class that has the largest $\mu_i c_i$ value.

In BERTSEKAS [10] convexity and a related concept called K-convexity of the optimal value function are used to derive structural properties in inventory control problems.

Application of the DPE procedure is usually limited to queueing systems where the state space has a small dimension. The main reason for this is the large number of different cases that have to be considered in the induction step of the algorithm, where one has to prove that properties of the value function are preserved by the DP operator T. The special cases typically occur at the boundaries of the state space

(see for instance HAJEK [40]). As the number of special cases is likely to explode for MDP's with a state space of higher dimensions, results deal mostly with one or two-dimensional problems. An attempt to unify the method for higher dimensions is proposed in BARTROLI AND STIDHAM [5].

3.3.4.2 Interchange Arguments

An alternative approach to prove structural properties of optimal policies is by using *interchange* or *sample path arguments.* This method bypasses dealing with the value function and the idea behind it is to prove the optimality of some policy by showing that it dominates any other policy in some sense of stochastic ordering. For this reason the method is also called the *stochastic dominance procedure.*

To prove the dominance of the optimal policy we have to prove that any other policy can be improved. The improvement is accomplished by interchanging action times, such that the newly constructed sequence of actions is in accordance with the structure of the optimal policy. As an example we can think of a scheduling problem in a queueing system, where decisions have to be made whether and when a server should be activated to serve customers. Suppose that we want to prove that an optimal policy is non-idling, i.e. the server is always activated when there is at least one customer in the queue. To prove this the sequence of actions of an idling policy should then be changed, by interchanging action times, to a policy that is non-idling, for instance by inserting an activation when the idling policy does not activate.

In the construction of a new policy by interchange, one often has to reorganise the underlying probability space to allow a comparison of the two policies for any realisation of the stochastic process. In some cases one has to construct equivalent stochastic processes that both have the same family of finite dimensional distribution functions and can be compared on the same probability space. One of the merits of the stochastic dominance procedure is that usually very little structure on the arrival and service processes has to be assumed, thus relaxing the exponential assumption that one often encounters in the DPE method.

Interchange arguments have been successfully applied in proving the optimality of the μc-rule (cf. BARAS *et al.* [3] and BUYUKKOC *et al.* [18]). Optimality of the μc-rule in tandem nodes, in queues with input from a network and in queues with feedback are discussed in NAIN [64] and NAIN *et al.* [67].

Interchange arguments are used in WALRAND [104] to prove the results of LIN AND KUMAR [60].

In BHATTACHARYA AND EPHREMIDES [12] a control problem for a queueing system with deadlines is considered. They consider a queueing network where sojourn times of customers must remain smaller than a randomly set deadline or else they are dropped from the system. The control algorithm has to specify which customer to serve next in order to minimise the fraction of lost customers. With interchange arguments it is shown to be optimal to serve the customer with the shortest time to expiration of its deadline.

3.3.4.3 Forward induction

The third method for the synthesis problem in stochastic control is called *forward induction.* It proceeds as follows. Assume that we have two policies $f, g \in \mathcal{U}$ and the

corresponding stochastic processes x_n, y_n on the same state space $\mathcal{S}$. If we want to prove that the rewards for policy f are larger than for g, then the idea is to construct a partial ordering $\rhd$ on $\mathcal{S}$ such that

(i) $x_n \rhd y_n$ implies $x_m \rhd y_m$ for all $m \geq n$

(ii) $x_n \rhd y_n$ implies

$$E_f\left[\sum_{n=0}^{\infty} \alpha^n r\big(x_n, f(x_n)\big)\right] \geq E_g\left[\sum_{n=0}^{\infty} \alpha^n r\big(y_n, f(y_n)\big)\right]$$

The partial ordering $\rhd$ is often formulated as a stochastic ordering, e.g. $x_n \rhd y_n$ if $P(x_n \leq a) \leq P(y_n \leq a)$ for all a. In addition it is sometimes required to construct equivalent processes $\tilde{x}_n$ and $\tilde{y}_n$ that have the same family of finite dimensional distribution functions and subsequently induce a partial ordering on $\tilde{x}_n$ and $\tilde{y}_n$. The main difficulty is to show that the ordering is preserved at each state transition.

Forward induction was used in EPHREMIDES *et al.* [32] in a routing problem with one arrival process and two identical queues. The problem is to find a routing policy to minimise the average total delay. It is shown that it is optimal to route to the shortest queue (*join-the-shortest-queue policy*) provided the queue lengths can be observed. An alternative proof for this problem can be found in WALRAND [105, Section 8.3].

4

An M/M/c Queueing Model

4.1 Introduction

In this chapter we discuss a queueing model for call request processing in a multiprocessor telephone switch. The aim of this investigation is to study the performance of a simple queueing model in which impatience of customers is explicitly modelled. We are interested in developing a model for an SPC that exhibits its typical overload behaviour, viz. the sharp decrease in the effective call handling capacity in case of overload. Another point in the investigation is to study the effect of an overload control mechanism that either may admit or not admit call requests.

Impatience of customers is modelled by setting deadlines for the service completion epochs. This deadline is set by the customer at the moment of his arrival and is specified relatively to his arrival epoch. Each customer who completes his service before his deadline has expired — a connection that is established before a subscriber looses his patience – contributes to the number of successful service completions. We are interested in developing an admission control scheme that can either admit or reject customers and that maximises the mean number of successful service completions per time unit. This quantity is usually referred to as successful throughput, goodput or effective call handling rate.

In the literature relatively little attention has been paid to queueing models with impatient customers. In Tran-Gia and van Hoorn [99] an M/G/1 queue with batch arrivals and service time discretisation is presented as a model for a switch. The successful throughput is expressed in the steady state probabilities and a suggestion for an overload control scheme is given. In Baccelli [1] a GI/G/1 queue is introduced with limitations on the waiting and sojourn times of customers. Stability conditions and functional equations for the distribution functions are derived. In Boel and van Schuppen [13, 14] models for call request processing and the stochastic control problem formulation for these models are presented.

This chapter is organised as follows. In Section 4.2 we introduce the queueing model and the formulation of the discounted reward control problem. In Section 4.3 we dis-

cuss the synthesis problem and introduce a new device for the derivation of structural properties of the optimal admission control law. We show that under very mild conditions the optimal admission policy is a threshold policy, i.e. new call requests are admitted only if the number of requests already being processed in the switch does not exceed a maximum value. In Section 4.4 we use the optimality of the subclass of threshold policies to derive a computational algorithm for the design problem of the discounted optimal threshold. We shall show in Section 4.5 that under the same condition that guarantees optimality of threshold policies, the optimal threshold is always finite and bounded as the discount rate decreases to zero. This boundedness is subsequently used in Section 4.6 to solve the average reward control problem. With the results of Section 4.5 threshold policies are shown to be optimal for this problem and an algorithm for the design problem of the average optimal threshold is presented. We conclude this chapter with some numerical examples and conclusions in Section 4.7.

4.2 Description of the model

In this chapter we consider a queueing model for call request processing in a telephone switch (see Figure 4.1). We assume that the customers, representing the call requests, are generated by a Poisson process with constant rate λ. The multi-processor switch is modelled by a set of c servers, and each server can work on one customer at a time. The service times of customers are independent and exponentially distributed with mean $1/\mu$.

Upon arrival a customer may be admitted to the switch or refused admission. If a customer is rejected, he is assumed lost to the system without influencing future arrivals. Specifically this means that we do not model retries of call requests that are rejected. If a customer is admitted he joins a waiting queue if no free server is available. If there is a free server he can start his service immediately. When a server completes the service of a customer, then that customer leaves the system. The server then fetches a new call request from the waiting queue, if any, in the order of arrival. In Kendall's notation (cf. Kendall [51]), this means that our model (when all customer would be admitted) is a *First Come First Served M/M/c queue.*

To expand this classical queueing model we now introduce a notion of impatience of customers. We assume that the i-th customer, arriving at time t_i has a deadline D_i associated with him, where $\{D_i\}_{i\in\mathbb{N}}$ is a sequence of non-negative independent and identically distributed random variables with distribution function F_D. Define $t_i + D_i$ as the *extinction time* of this particular customer. If a customer is admitted he will complete his service after some time S_i, his sojourn time. The service completion is considered to be successful if it takes place before the extinction time, i.e. if $t_i + S_i \leq t_i + D_i$. We are interested in the fraction of customers that complete their services before their extinction times and wish to determine an admission policy that maximises this fraction. This fraction multiplied by the arrival rate is usually referred to as *goodput*, the throughput of 'good' or 'successful' customers. The decision to be made by the control algorithm is to reject or admit new arriving customers and the information for the algorithm is the complete history of the arrival and queueing processes.

Two observations ought to be made here about our modelling of customers' impatience. First we assume that a customer always waits for the completion of his service, irrespective of whether his deadline has already expired. In queueing terminology this

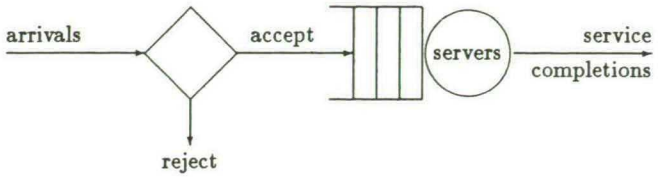

FIGURE 4.1. An M/M/c queueing model for an SPC switch.

means that a customer is not allowed to *balk*, i.e. leave the queue before his service is completed. This accounts for the observation in a real telephone switch where even abandoned call requests — call requests of which the deadlines have expired — require service from a processor.

Second we compare the extinction time with the end and not with the beginning of service. This agrees with the fact that subscribers are usually not aware of the start of service but only of its completion, since that instant coincides with the start of a connection. CPU's, however, usually work in a time-sharing fashion and thus service starts immediately after admittance. In this respect we have to admit that the M/M/c FCFS queue is a crude simplification. Since our primary concern is to develop an analytically tractable queueing model that exhibits the typical SPC's overload behaviour, we have chosen to consider only the M/M/c FCFS queue.

The optimisation problem for this queueing model can be translated into the framework of MDP's, as described in Section 3.3.2. The state of the MDP should be constructed such that it comprises all information necessary to determine the future evolution of the process plus the information for the control algorithm to make a decision on admittance of any new customers. Since both the interarrival and service times are exponentially distributed, the number of customers present — both in service and waiting — completely determines (in a Markovian sense) the future of the queueing system. We shall refer to this number as the *queue length*. To provide information on the state of the queue to the control algorithm we can choose between two alternatives. The first approach is to augment the queue length with a stochastic variable that can have the value 0 or 1 to indicate the absence or presence of a customer seeking admittance, respectively. For this option the decision available is to admit or reject a customer, provided there is one requesting to be admitted. In the second approach, we omit this extra variable and formulate the decision as: reject or admit a customer if there will be an arrival before the next service completion. For this option the momentary queue length provides all the necessary information due to the memoryless property of the interarrival time distribution. Since one can easily prove that both approaches lead to the same MDP, we shall restrict our attention to the second approach. This means that we can represent the state space of the MDP as $\mathcal{S} = \mathbb{N}$ with $n \in \mathcal{S}$ representing the queue length n.

If the queue length is n, then the time until the next service completion is exponentially distributed with rate $\mu_n := \mu \min(c, n)$. The sets of admissible actions $\mathcal{A}_n$ in state n are the same for all $n \in \mathbb{N}$ and can be represented as $\mathcal{A}_n = \{0, 1\}$. Taking action $a = 0$ or 1 corresponds to rejection or admission of a new arriving customer, if it occurs before the next service completion. Therefore if we take action a when the queueing process is in state n, then the time until the next state transition is expo-

nentially distributed with intensity $\gamma(n,a) = \mu_n + a\lambda$, $n \in \mathbf{N}$, $a \in \{0,1\}$. A uniform upper bound for this rate is $\Gamma = c\mu + \lambda$. Due to Remark 3.3.15 we can choose this upper bound as the constant and state-independent transition rate of the process, thereby introducing dummy transitions. Furthermore we can assume without loss of generality that $\Gamma = 1$, since that can be accomplished by appropriately scaling the time axis. Doing this we get for the transition probabilities from state n to m

$$p(n,a,m) = \begin{cases} \dfrac{\mu_n}{\Gamma}, & \text{if } m = n-1, \\ \dfrac{\mu_c - \mu_n + (1-a)\lambda}{\Gamma}, & \text{if } m = n, \\ \dfrac{a\lambda}{\Gamma}, & \text{if } m = n+1. \end{cases}$$

Next in the formulation of the MDP is the reward $r(n,a)$ that is received when in state n action a is chosen. Since we want to maximise the fraction of customers who complete their services before their extinction times, we gain a reward of 1 if we admit a customer whose sojourn time will be smaller than his deadline and we receive a reward of 0 otherwise. Assume now that the state is n and we choose action $a = 1$. If the next event is an arrival — this happens with probability λ — the customer is admitted, joins the queue and increases the queue length to $n+1$. We now define a function $g : \mathbf{N} \to \mathbf{R}_+$ that we use to construct a reward function r.

DEFINITION 4.2.1 *Let $g(n)$, $n \geq 1$, be the probability that a customer who is admitted to the queue, makes his deadline when the queue length, including himself, at the time of arrival is n.*

With this definition of g the reward $r(n,a)$, gained at the moment of arrival, is thus $r(n,a) = a\lambda g(n+1)$. Although the 'fraction' of customers that meet their deadline is an average reward criterion, we shall first address the discounted reward control problem. Define the total return $W_u^T : \mathbf{N} \to \mathbf{R}$ gained under policy $u \in \mathcal{U}$ over horizon length $T > 0$ as

$$W_u^T(n) = E_u\left[\sum_{\{n \mid t_n \leq T\}} r(X_{t_n-}, a_{t_n-}) \;\middle|\; X_0 = n \right], \qquad n \in \mathbf{N}. \tag{4.1}$$

Here t_n denotes the n-th transition time of the queueing process, X_{t_n} the queue length at time t_n, and a_{t_n} the decision according to policy u at time t_n. Let $\alpha > 0$ be a discount rate. Recall from Chapter 3 that the infinite horizon discounted reward control problem (P_α) is

(P_α): maximise $V_{u,\alpha}(n)$ over $\mathcal{U}$ for all $n \in \mathbf{N}$,

where the total discounted return $V_{u,\alpha} : \mathbf{N} \to \mathbf{R}$ under policy $u \in \mathcal{U}$ with initial state n is defined as

$$V_{u,\alpha}(n) = \lim_{T \to \infty} \int_0^T e^{-\alpha t} d_t W_u^t(n), \qquad n \in \mathbf{N}. \tag{4.2}$$

If we want to solve the infinite horizon discounted reward control problem, we know from Theorem 3.3.13 that we have to solve the DPE equations:

$$\alpha V(n) = \max_{u_n \in \{0,1\}} \Big\{ u_n \lambda g(n+1) + \mu_n \big[V(n-1) - V(n)\big] + u_n \lambda \big[V(n+1) - V(n)\big] \Big\},$$

for all $n \in \mathbb{N}$, or equivalently

$$\alpha V(n) = \mu_n \big[V(n-1) - V(n)\big] + \max_{u_n \in \{0,1\}} u_n \lambda \big[V(n+1) - V(n) + g(n+1)\big],$$

for all $n \in \mathbb{N}$, with $V(-1) = 0$. Since $g(n)$, $n \in \mathbb{N}$ denotes a probability, the reward function $r(n,a) = a\lambda g(n+1)$ is non-negative and bounded. We know from Theorem 3.3.13 that the policy u^* that is optimal for (P_α), is stationary and is given by the solution of the functional equations

$$\alpha V(n) = \mu_n \big[V(n-1) - V(n)\big] + u_n^* \lambda \big[V(n+1) - V(n) + g(n+1)\big], \tag{4.3}$$

for $n \in \mathbb{N}$, and u_n^*, the optimal decision in state n, is

$$u_n^* = \begin{cases} 0, & \text{if } V(n+1) - V(n) + g(n+1) \leq 0, \\ 1, & \text{if } V(n+1) - V(n) + g(n+1) > 0. \end{cases} \tag{4.4}$$

In the next section we shall present conditions on the function g, that guarantee that u^* is of the form $u_n^* = 1(n < L)$ for some $L \in \mathbb{N}$. Policies of this form are called *threshold policies*, since they result in admittance of all customers up to the threshold L and no further admittance until the number of customers decreases again. We shall also present examples of deadline distributions, that make g have these desired properties.

We conclude this section with a remark about the action space. In our model the admission policy can either admit or not admit arriving customers. We can allow this decision to be randomised by extending the action space to $\mathcal{A} = [0,1]$. In this case a decision $a \in [0,1]$ means that with probabilities a a new customer is admitted and with probability $1-a$ he is rejected. Since this formulation is not in accordance with the framework that we introduced in Section 3.3 — the action space is not countable — we shall not discuss this problem. Furthermore, since the reward function $r(n,a)$ is linear in a, even with randomised admission policies the optimal policy would always be non-randomised (cf. BRÉMAUD [16]).

4.3 Optimality of threshold policies

In this section we shall present sufficient conditions on the deadline distribution under which the optimal admission policy is a threshold policy. We shall prove that these conditions are for instance met when the deadlines are deterministic or Erlang distributed (including of course the exponential distribution). The method of proof for the optimality of threshold policies is new and derives the structural properties of the optimal value function directly from the DP equations for the infinite horizon. Specifically this means that we do not need to prove structural properties for the optimal finite-horizon value function like in the backward recursion or sample path method.

The method of proof was first used in DE WAAL [100] for a queueing model similar to the one that is discussed in this chapter. In [100] rewards were given for service completions and the decision to admit or reject customers was allowed to be randomised. Parts of the adaptation of that model to the control problem in this chapter were obtained in cooperation with Philippe Nain, Don Towsley and Hans Blanc.

We start with a formulation of the sufficient conditions for the function g.

DEFINITION 4.3.1 (MODIFIED UNIMODALITY)
A function $g : \mathbf{N} \to \mathbf{R}_+$ *is called* modified unimodal *with parameter* ψ, $0 \leq \psi < 1$, *if it satisfies the following conditions:*

(i) $g(n)\mu_n \leq g(n+1)\mu_{n+1}$ *for* $n < c$.

(ii) $g(n+1) \leq \psi g(n)$ *for* $n \geq c$.

This definition is related to the classical notion of *unimodality* of a function, that is defined as the existence of an $l \in \mathbf{N}$ such that

(i) $g(n) \leq g(n+1)$ for $n < l$

(ii) $g(n) \geq g(n+1)$ for $n \geq l$.

Comparing this with Definition 4.3.1 (i) we see that part (i) of unimodality is a slightly stronger condition, whereas Definition 4.3.1 (ii) is stronger than part (ii) of unimodality.

The following lemma shows that modified unimodality is a sufficient condition on the reward function g for the optimality of threshold policies.

LEMMA 4.3.2 *Let* g *be a modified unimodal function with parameter* ψ. *The optimal value function* V^* *for the DP equations (4.3) and the corresponding optimal policy* u^*, *given by* $u_n^* = 1(V^*(n+1) - V^*(n) + g(n+1) > 0)$, *satisfy the following conditions.*

(i) $u_n^* = 1$ *for* $0 \leq n < c$.

(ii) if $\alpha > 0$ *is such that* $c\mu/(c\mu + \alpha) > \psi$ *and if there is an* $l \in \mathbf{N}$ *such that* $u_l^* = 0$, *then* $u_n^* = 0$ *for all* $n \geq l$.

Two remarks are in order here. Properties (i) and (ii) indeed guarantee that u^* is a threshold policy, since $u_n^* = 1(n < L^*)$, with $L^* = \inf\{\, l \mid u_l^* = 0 \,\}$. Included here is the case $L^* = \infty$, though we shall show in Section 4.5 that for modified unimodal g this possibility is excluded.

The second remark concerns the condition $c\mu/(c\mu+\alpha) > \psi$. This is not a restrictive condition, since we are primarily interested in the optimal policy for the average reward criterion. Recall from Section 3.3.2.2 that this problem can be treated as the limit of problem (P_α) as $\alpha \downarrow 0$.

PROOF OF LEMMA 4.3.2. Let V^* be the solution of the DP equations (4.3). Since the reward function $r(n,a)$ is bounded, we know from the Dynamic Programming Theorem 3.3.13 that V^* is the unique bounded solution of these equations. Also let u^* be the optimal stationary policy, corresponding to V^*, as defined in (4.4). Define $x^* : \mathbf{N} \to \mathbf{R}$ as

$$x^*(n) = \begin{cases} V^*(0), & \text{if } n = 0, \\ V^*(n) - V^*(n-1), & \text{if } n > 0. \end{cases}$$

We can rewrite the DP equations (4.3) and (4.4) as

$$0 = -\alpha \sum_{i=0}^{n} x^*(i) - \mu_n x^*(n) + \lambda u_n^* \big[x^*(n+1) + g(n+1) \big], \qquad n \in \mathbf{N}, \tag{4.5}$$

and

$$u_n^* = 1\big(x^*(n+1) + g(n+1) > 0\big), \qquad n \in \mathbf{N}. \tag{4.6}$$

Note that

$$u_n^*\big[x^*(n+1) + g(n+1)\big] \geq 0, \qquad n \in \mathbf{N}. \tag{4.7}$$

We now prove (i) by induction in n.
Initial step. Substituting $n = 0$ and $n = 1$ into (4.5) yields

$$0 = -\alpha x^*(0) + \lambda u_0^*\big[x^*(1) + g(1)\big], \tag{4.8}$$

$$0 = -\alpha\big[x^*(0) + x^*(1)\big] - \mu_1 x^*(1) + \lambda u_1^*\big[x^*(2) + g(2)\big]. \tag{4.9}$$

Subtracting (4.8) from (4.9) yields

$$\lambda u_0^*\big[x^*(1) + g(1)\big] - \lambda u_1^*\big[x^*(2) + g(2)\big] = -(\alpha + \mu_1)x^*(1). \tag{4.10}$$

If we assume that $u_0^* = 0$, then we see that the left-hand side of (4.10) is non-positive due to (4.7). However, because $u_0^* = 0$, (4.6) implies that $x^*(1) \leq -g(1) < 0$, and so the right-hand side of (4.10) must be strictly positive, which implies a contradiction. We may conclude that $u_0^* = 1$.
Induction step. Assume now that $u_0^* = u_1^* = \cdots = u_{l-1}^* = 1$ for $l < c - 1$ and let us show that $u_l^* = 1$. Substituting $n = l$ and $n = l + 1$ into (4.5) and subtracting the equation for $n = l$ from the equation for $n = l + 1$ yields

$$\lambda u_{l+1}^*\big[x^*(l+2) + g(l+2)\big] - \lambda u_l^*\big[x^*(l+1) + g(l+1)\big] =$$

$$(\alpha + \mu_{l+1})x^*(l+1) - \mu_l x^*(l).$$

Let us assume now that $u_l^* = 0$, then this implies

$$(\alpha + \mu_{l+1})x^*(l+1) - \mu_l x^*(l) \geq 0, \tag{4.11}$$

since clearly $u_{l+1}^*[x^*(l+2) + g(l+2)] \geq 0$. Adding and subtracting the same terms shows this to be equivalent to

$$(\alpha + \mu_{l+1})\big[x^*(l+1) + g(l+1)\big] - \mu_l\big[x^*(l) + g(l)\big]$$

$$-g(l+1)\big[\alpha + \mu_{l+1}\big] + \mu_l g(l) \geq 0.$$

Note that $x^*(l+1) + g(l+1) \leq 0$ (by the assumption $u_l^* = 0$), $x^*(l) + g(l) > 0$ (since $u_{l-1}^* = 1$ by the induction assumption) and $g(l)\mu_l - g(l+1)\mu_{l+1} \leq 0$ (since g is modified unimodal). The left-hand side of (4.11) is thus strictly negative, which gives a contradiction. We may conclude that $u_l^* = 1$.

We shall prove part (ii) by constructing a bounded real-valued function that solves the DP equations. By Theorem 3.3.13, that states that there exists exactly one bounded solution, we may then conclude that this constructed value function is indeed the correct one.

Let $\alpha > 0$ satisfy $c\mu/(c\mu + \alpha) > \psi$. Furthermore let $l \in \mathbf{N}$, $l \geq c$, be such that $u_l^* = 0$. This implies that $x^*(l+1) \leq -g(l+1)$. Define $x : \mathbf{N} \to \mathbf{R}$ as

$$x(n) = \begin{cases} x^*(n), & \text{if } 0 \le n \le l, \\ -\dfrac{\alpha}{\alpha + \mu_n} \sum_{i=0}^{n-1} x(i), & \text{if } n > l. \end{cases}$$

Note that $x(n)$, $n > l$, is the recursion we get from (4.5), if we choose $u_n^* = 0$ for $n > l$. We prove that $x(n) \le -g(n)$ for $n > l$ by induction in n.
Initial step. Let $n = l + 1$. From the definition of x and x^* we have

$$x(l+1) = -\frac{\alpha}{\alpha + \mu_{l+1}} \sum_{i=0}^{l} x(i)$$

$$= -\frac{\alpha}{\alpha + \mu_{l+1}} \sum_{i=0}^{l} x^*(i).$$

Taking $n = l + 1$ in (4.5) shows that this is equal to

$$\left\{(\alpha + \mu_{l+1})x^*(l+1) - \lambda u_{l+1}^* \left[x^*(l+2) + g(l+2)\right]\right\} / (\alpha + \mu_{l+1})$$

$$\le x^*(l+1)$$

$$\le -g(l+1).$$

The last two steps follow from (4.7) and the fact that $x^*(l+1) \le -g(l+1)$.
Induction step. Assume that $x(i) \le -g(i)$ for $l + 1 \le i \le n$. We shall show that $x(n+1) \le -g(n+1)$. By definition of $x(n)$ we have

$$x(n+1) = \left[-\alpha x(n) - \alpha \sum_{i=0}^{n-1} x(i)\right] / (\alpha + \mu_{n+1})$$

$$= -\left[\alpha x(n) - (\alpha + \mu_n)x(n)\right]/(\alpha + \mu_{n+1})$$

$$= \mu_n x(n)/(\alpha + \mu_{n+1})$$

$$\le -\mu_c g(n)/(\alpha + \mu_c)$$

$$\le -g(n+1).$$

Here we use the induction assumption, the modified unimodality of g, the fact that $\mu_n = \mu_c$ for $n \ge c$ and the condition that $c\mu/(c\mu + \alpha) > \psi$.

Define now $V(n) = \sum_{i=0}^{n} x(i)$, $n \in \mathbb{N}$. We shall show that $V(n)$ is bounded. Since by construction $V(n) = V^*(n)$, for $0 \le n \le l$, $V(n)$ is bounded for $0 \le n \le l$. The boundedness for $n > l$ follows from the definition of x:

$$V(n) - V(n-1) = -\frac{\alpha}{\alpha + \mu_n} V(n-1), \qquad n > l,$$

or equivalently

$$V(n) = \frac{\mu_n}{\alpha + \mu_n} V(n-1), \qquad n > l.$$

Now define $u_n = 1(V(n+1) - V(n) + g(n+1) > 0)$. We thus have constructed a solution to equations (4.3) and (4.4). Since V is bounded, it must be the unique solution to these DP equations. Moreover V satisfies (i) and (ii) by construction. □

The methodology used in the proof of Lemma 4.3.2 does not fall into any of the categories that were described in Section 3.3.4. Therefore we remark that the importance of the lemma lies not as much in the optimality of threshold policies (which is what we would intuitively expect) but rather in the method of proof. We conjecture that it is possible to extend the method to more complicated queueing systems.

We conclude this section with some examples for the deadline distribution, that make the reward function g modified unimodal. Recall that $g(n)$ was defined as the probability that an arriving customer makes his deadline when the queue length after his arrival, including himself, is n.

PROPOSITION 4.3.3 *If the deadlines of customers are deterministic, then g is modified unimodal.*

PROOF. See Appendix B.

The next class of distribution functions comprises a large class of absolutely continuous distributions. For this we need the following definition (cf. STOYAN [93] and BARLOW AND PROSCHAN [4]).

DEFINITION 4.3.4 (FAILURE RATE)
If a non-negative random variable has a distribution function $F : \mathbf{R}_+ \to [0,1]$ and probability density $f : \mathbf{R}_+ \to \mathbf{R}_+$, then the failure rate *(also called* hazard rate*) $h : \{ t \in \mathbf{R}_+ \mid F(t) < 1 \} \to \mathbf{R}_+$ is defined as*

$$h(t) = \frac{f(t)}{1 - F(t)}.$$

If there exists a constant $\Psi > 0$ such that

$$\inf_{\{t \mid F(t) < 1\}} h(t) > \Psi,$$

then we say that the random variable has a failure rate that is bounded away from 0 by Ψ.

PROPOSITION 4.3.5 *If the deadlines of customers have a failure rate that is bounded away from 0 by Ψ, then the reward function g is modified unimodal with parameter $\psi = c\mu/(c\mu + \Psi)$.*

PROOF. See Appendix B

The set of distribution functions with a failure rate that is bounded away from 0, comprises a large class of distributions, e.g. the exponential distribution and subsets of the class of Gamma distributions and truncated normal distributions (cf. BARLOW AND PROSCHAN [4, Section 3.5]). Unfortunately Erlang distributions do not fall in this category, since their failure rates $h(t)$ tend to 0 as $t \downarrow 0$. We can prove, however, that for Erlang distributed deadlines, too, g is modified unimodal.

PROPOSITION 4.3.6 *If the deadlines of customers have an Erlang distribution, then the reward function g is modified unimodal.*

PROOF. See Appendix B.

4.4 Design of the optimal threshold

In this section we present an algorithm that efficiently computes the optimal threshold policy for problem (P_α) as described in Section 4.3. Let g again denote a modified unimodal function with parameter ψ and assume that α is sufficiently close to 0 such that $\mu_c/(\alpha+\mu_c) > \psi$. If we choose the lump reward $r : \mathbf{N} \times \{0,1\} \to \mathbf{R}_+$ as $r(n,a) = a\lambda g(n+1)$, then we know from Lemma 4.3.2 that the optimal admission policy u^* is a threshold policy, so $u^*_n = 1(n < k^*)$ for some $k^* \in \mathbf{N}$.

Let $k \in \mathbf{N}$. If we want to test whether $k^* = k$, then we have to compute the discounted value function V_k for policy u^k, where $u^k_n = 1(n < k)$, from the set of equations

$$\begin{aligned} 0 = &-\alpha V_k(n) + \mu_n\big[V_k(n-1) - V_k(n)\big] \\ &+\lambda 1(n<k)\big[V_k(n+1) - V_k(n) + g(n+1)\big], \quad n \in \mathbf{N}. \end{aligned} \tag{4.12}$$

If k is indeed the optimal threshold, then V_k has to be consistent with u^k, i.e.

$$V_k(n+1) - V_k(n) + g(n+1) > 0 \text{ iff } n < k. \tag{4.13}$$

If V_k is consistent, then we have found the optimal threshold k^*, otherwise we have to try another value. The computation of V_k from (4.12) and the test of its consistency with u^k can be performed quite efficiently as follows.

We rewrite the first $k+1$ equations of (4.12) in the matrix notation

$$A_k \mathbf{V}_k = \mathbf{g}_k, \tag{4.14}$$

where

$$\mathbf{V}_k = \big[V_k(0), V_k(1), \ldots, V_k(k-1), V_k(k)\big]^T, \tag{4.15}$$

$$\mathbf{g}_k = \lambda\big[g(1), g(2), \ldots, g(k), 0\big]^T, \tag{4.16}$$

and $A_k : \mathbf{N}_{k+1} \times \mathbf{N}_{k+1} \to \mathbf{R}$ is the matrix

$$\begin{bmatrix} \alpha+\lambda & -\lambda & & & \\ -\mu_1 & \alpha+\lambda+\mu_1 & -\lambda & & \\ & \cdot & \cdot & \cdot & \\ & & -\mu_{k-1} & \alpha+\lambda+\mu_{k-1} & -\lambda \\ & & & -\mu_k & \alpha+\mu_k \end{bmatrix}, \tag{4.17}$$

with the rows and columns numbered from 0 to k. The matrix equation (4.14) can be solved by standard *LU-decomposition* (cf. Golub and Van Loan [38]), using the diagonal entries of A_k as the pivots starting with column 0. Let $A^n_k = M_n M_{n-1} \ldots M_0 A_k$, $0 \le n \le k-1$, denote the matrix that is obtained after the n-th step of the LU-decomposition, so A^n_k has all the entries below the diagonal in the columns 0 to n eliminated. The matrix A^{k-1}_k has the following form

$$\begin{bmatrix} A_k^{k-1}[0] & -\lambda & & & & \\ & A_k^{k-1}[1] & -\lambda & & & \\ & & \ddots & \ddots & & \\ & & & A_k^{k-1}[k-1] & -\lambda & \\ & & & & A_k^{k-1}[k] & \end{bmatrix}. \tag{4.18}$$

Let $\mathbf{g}_k^{k-1} = M_{k-1}M_{k-2}\dots M_0\mathbf{g}_k = \left[g_k^{k-1}(1),\dots,g_k^{k-1}(k+1)\right]^T$. Since the matrices M_n are non singular, V_k is also the solution to the matrix equation

$$A_k^{k-1}\mathbf{V}_k = \mathbf{g}_k^{k-1}.$$

From (4.18) we can immediately solve for $V_k(k)$:

$$V_k(k) = \mathbf{g}_k^{k-1}(k)/A_k^{k-1}[k].$$

To test whether V_k is consistent with u^k we have to compute $V_k(k+1)$. Substituting $n = k+1$ in (4.12) yields

$$V_k(k+1) = \frac{\mu_{k+1}}{\alpha+\mu_{k+1}}V_k(k).$$

If the solution for V_k is consistent with u^k, i.e.

$$V_k(k+1) - V_k(k) + g(k+1) \le 0,$$

then k is the optimal threshold and the algorithm terminates. If V_k and u^k are not consistent, then clearly k is not the optimal threshold and we have to try another value for k^*. An obvious next guess would be $k+1$, meaning that we have to solve the matrix equation $A_{k+1}\mathbf{V}_{k+1} = \mathbf{g}_{k+1}$. Following the same steps in the LU-decomposition as for A_k it is easy to show that A_{k+1}^{k-1} is almost identical to A_k^{k-1}:

$$\begin{bmatrix} A_k^{k-1}[0] & -\lambda & & & & \\ & A_k^{k-1}[1] & -\lambda & & & \\ & & \ddots & \ddots & & \\ & & & A_k^{k-1}[k-1] & -\lambda & \\ & & & & A_k^{k-1}[k]+\lambda & -\lambda \\ & & & & -\mu_{k+1} & \alpha+\mu_{k+1} \end{bmatrix}, \tag{4.19}$$

and $\mathbf{g}_{k+1}^{k-1} = \left[g_k^{k-1}(1),\dots,g_k^{k-1}(k), g_k^{k-1}(k+1)+g(k+1)\right]^T$. A_{k+1}^k can thus easily be computed from the intermediate result A_k^{k-1} (as well as $\mathbf{g}_{k+1}^k$ from $\mathbf{g}_k^{k-1}$). This enables us to solve the DP equations through one set of linear equations by incorporating the inequality check (4.13) into the LU-decomposition. The algorithm can be summarised as follows

ALGORITHM 4.4.1

(i) Set $k := 1$.

(ii) Compute $A_k^{k-1}[k]$ and $g_k^{k-1}(k)$. Determine $V_k(k)$ and $V_k(k+1)$ from

$$V_k(k) = \frac{g_k^{k-1}(k)}{A_k^{k-1}[k]}$$

and

$$V_k(k+1) = \frac{\mu_{k+1}}{\alpha + \mu_{k+1}} V_k(k).$$

(iii) If $V_k(k+1) - V_k(k) + g(k+1) \leq 0$, then stop, otherwise increase k by 1 and proceed with step *(ii)*.

As we know from Lemma 4.3.2 that the optimal threshold is at least c, the number of servers, we could let the search procedure for k^* start at the value c. We have not included this feature in the Algorithm 4.4.1, since we have to perform the LU-decomposition for the first c columns of the matrices A_k anyway.

In the next section we shall show that if g is modified unimodal, then the algorithm terminates within a finite number of iterations.

4.5 FINITENESS OF THE OPTIMAL THRESHOLD

In this section we shall show that if the function g is modified unimodal, then for sufficiently small $\alpha > 0$ the α-discounted optimal threshold is finite. Moreover we shall show that the optimal thresholds are bounded as α tends to 0 and use this result in Section 4.6 to prove that the average reward control problem can be solved by taking the limit of the discounted reward control problem. An immediate consequence of this convergence will be the optimality of threshold policies for the average reward control problem.

Let $k \in \mathbb{N}$, $k \geq c$. Assume that g is a modified unimodal function with parameter ψ. Take $\alpha > 0$ such that $\mu_c/(\alpha + \mu_c) > \psi$. Let u^k denote the stationary threshold policy with threshold k, i.e. $u_n^k = 1(n < k)$, $n \in \mathbb{N}$. We define V_k to be the discounted value function for policy u^k and x_k the first-order difference of V_k, i.e.

$$x_k(n) = \begin{cases} V_k(0), & \text{if } n = 0, \\ V_k(n) - V_k(n-1), & \text{if } n > 0. \end{cases}$$

The values of $x_k(n)$, $n \in \mathbb{N}$ can be determined as the solution of the following set of linear equations.

$$0 = -\alpha x_k(0) + \lambda\big[x_k(1) + g(1)\big], \tag{4.20a}$$

$$0 = -\alpha \sum_{i=0}^{n} x_k(i) + \lambda\big[x_k(n+1) + g(n+1)\big] - \mu_n x_k(n), \qquad 1 \leq n < k, \tag{4.20b}$$

$$0 = -\alpha \sum_{i=0}^{n} x_k(i) - \mu_n x_k(n), \qquad n \geq k. \tag{4.20c}$$

The first $k+1$ equations can be represented in matrix notation as

$$B_k \mathbf{x}_k = \mathbf{g}_k, \tag{4.21}$$

where

$$\mathbf{x}_k = \left[x_k(0), x_k(1), \ldots, x_k(k-1), x_k(k)\right]^T, \tag{4.22}$$

$$\mathbf{g}_k = \lambda\left[g(1), g(2), \ldots, g(k), 0\right]^T, \tag{4.23}$$

and $B_k : \mathbf{N}_{k+1} \times \mathbf{N}_{k+1} \to \mathbf{R}$ is given by

$$\begin{bmatrix} \alpha & -\lambda & & & & & \\ \alpha & \alpha+\mu_1 & -\lambda & & & & \\ \alpha & \alpha & \alpha+\mu_2 & -\lambda & & & \\ . & . & . & . & . & & \\ . & . & . & . & . & . & \\ \alpha & . & . & . & \alpha & \alpha+\mu_{k-1} & -\lambda \\ \alpha & . & . & . & . & \alpha & \alpha+\mu_k \end{bmatrix}. \tag{4.24}$$

For convenience we number the rows and columns of B_k from 0 to k. Note that we have $k+1$ unknown variables and $k+1$ equations. We let Y_k denote the matrix that is obtained by replacing the last row of B_k by the $k+1$-dimensional vector $[1, 1, \ldots, 1, 1]$. With straightforward computation it is easy to show that we have the following recursion for the determinants of B_k and Y_k:

$$\det(B_k) = (\alpha+\mu_k)\det(B_{k-1}) + \alpha\lambda\det(Y_{k-1}), \qquad k \geq 2, \tag{4.25}$$

$$\det(Y_k) = \det(B_{k-1}) + \lambda\det(Y_{k-1}), \qquad k \geq 2, \tag{4.26}$$

with the initial values $\det(B_0) = \alpha$, $\det(B_1) = \alpha(\alpha+\lambda+\mu_1)$, $\det(Y_0) = 1$ and $\det(Y_1) = \alpha+\lambda$. As an immediate result we can show by induction that $\det(B_k) > 0$, so the matrix equation (4.21) has a unique solution.

Assume now that the optimal threshold is at least k. Analogously to the method that was used in Section 4.4 the optimal threshold is equal to k if the solution x_k of (4.21) is consistent with the policy u^k, i.e. if

$$x_k(i) + g(i) > 0, \qquad i = 0, \ldots, k, \tag{4.27}$$

$$x_k(k+1) + g(k+1) \leq 0. \tag{4.28}$$

The value of $x_k(k+1)$ can be expressed in $x_k(k)$ as follows. From (4.20c) we have

$$0 = -\alpha \sum_{i=0}^{k} x_k(i) - \mu_k x_k(k), \tag{4.29a}$$

$$0 = -\alpha \sum_{i=0}^{k+1} x_k(i) - \mu_{k+1} x_k(k+1). \tag{4.29b}$$

Subtracting (4.29a) from (4.29b) yields

$$x_k(k+1) = \frac{\mu_k}{\alpha + \mu_{k+1}} x_k(k).$$

The inequality (4.28) then becomes equivalent to

$$\frac{\mu_k}{\alpha + \mu_{k+1}} x_k(k) + g(k+1) \leq 0. \tag{4.30}$$

We shall show that we can reformulate (4.30) as a condition on the function g. For this purpose we introduce the following notation. Let $D_k^{(i,j)}$, $i, j = 0, \ldots, k$, denote the $N_k \times N_k$ matrix that is obtained from B_k by deleting the i-th row and j-th column.

LEMMA 4.5.1 *For all* $i = 0, \ldots, k-2$

$$\det\left(D_k^{(i,k)}\right) = -\mu_{k-1} \det\left(D_{k-1}^{(i,k-1)}\right). \tag{4.31}$$

PROOF. Let $i = 0, \ldots, k-2$. $D_k^{(i,k)}$ has the following form:

$$\begin{bmatrix}
\alpha & -\lambda & & & & & & & \\
\alpha & \alpha+\mu_1 & -\lambda & & & & & & \\
\cdot & \cdot & \cdot & \cdot & & & & & \\
\alpha & \cdot & \alpha & \alpha+\mu_{i-1} & -\lambda & & & & \\
\alpha & \cdot & \cdot & \cdot & \alpha & \alpha+\mu_{i+1} & -\lambda & & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
\alpha & \cdot & \cdot & \cdot & \cdot & \cdot & \alpha & \alpha+\mu_{k-2} & -\lambda \\
\alpha & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \alpha & \alpha+\mu_{k-1} \\
\alpha & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \alpha & \alpha
\end{bmatrix}.$$

In the computation of the determinant of this matrix we are allowed to subtract the last row from the next to last one, yielding

$$\begin{bmatrix}
\alpha & -\lambda & & & & & & & \\
\alpha & \alpha+\mu_1 & -\lambda & & & & & & \\
\cdot & \cdot & \cdot & \cdot & & & & & \\
\alpha & \cdot & \alpha & \alpha+\mu_{i-1} & -\lambda & & & & \\
\alpha & \cdot & \cdot & \cdot & \alpha & \alpha+\mu_{i+1} & -\lambda & & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
\alpha & \cdot & \cdot & \cdot & \cdot & \cdot & \alpha & \alpha+\mu_{k-2} & -\lambda \\
0 & 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & \mu_{k-1} \\
\alpha & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \alpha & \alpha
\end{bmatrix}.$$

From this it is immediate that $\det(D_k^{(i,k)}) = -\mu_{k-1}\det(D_{k-1}^{(i,k-1)})$. □

With Lemma 4.5.1 we can write down the following recursion for $x_k(k)$.

LEMMA 4.5.2 *For all* $k \geq 1$

$$x_k(k)\det(B_k) = -\alpha\lambda\sum_{i=1}^{k} M(k,i)\,g(i)\det(Y_{i-1}), \tag{4.32}$$

where

$$M(k,i) = \prod_{j=i}^{k-1}\mu_j, \qquad i = 1,\ldots,k. \tag{4.33}$$

PROOF. It follows from Cramer's rule that

$$x_k(k)\det(B_k) = \lambda\sum_{i=0}^{k-1} g(i+1)\,(-1)^{k+i}\det(D_k^{(i,k)})$$

With Lemma 4.5.1 this yields for $k \geq 1$

$$x_k(k)\det(B_k) = \mu_{k-1}x_{k-1}(k-1)\det(B_{k-1}) - \alpha\lambda\,g(k)\det(Y_{k-1}). \tag{4.34}$$

Recursive application of (4.34) yields (4.32). □

LEMMA 4.5.3 *For* $k \geq 2$,

$$\det(B_k) = \alpha\lambda\sum_{i=1}^{k-1} K(k,i)\det(Y_i) + K(k,0)\det(B_1), \tag{4.35}$$

where

$$K(k,i) = \prod_{j=i+2}^{k}(\alpha+\mu_j), \qquad i = 1,\ldots,k-1. \tag{4.36}$$

PROOF. Recursively apply (4.25). □

With the results of Lemma 4.5.2 and 4.5.3 and the fact that $\det(B_k) > 0$ the inequality (4.30) for $k \geq c$ becomes equivalent to

$$g(k+1)\det(B_k) \leq -\frac{\mu_k}{\alpha+\mu_{k+1}}x_k(k)\det(B_k) \tag{4.37}$$

$$\begin{aligned} \Leftrightarrow\ & g(k+1)\left[\alpha\lambda\sum_{i=1}^{k-1}K(k,i)\det(Y_i) + K(k,0)\det(B_1)\right] \\ & \leq \alpha\lambda\frac{\mu_k}{\alpha+\mu_{k+1}}\sum_{i=1}^{k}M(k,i)\,g(i)\det(Y_{i-1}) \end{aligned} \tag{4.38}$$

$$\begin{aligned} \Leftrightarrow\ & g(k+1)\left[\alpha\lambda\sum_{i=2}^{k}K(k,i-1)\det(Y_{i-1}) + K(k,0)\det(B_1)\right] \\ & \leq \alpha\lambda\frac{\mu_k}{\alpha+\mu_{k+1}}\sum_{i=1}^{k}M(k,i)\,g(i)\det(Y_{i-1}). \end{aligned} \tag{4.39}$$

LEMMA 4.5.4 *If the function g is modified unimodal with parameter ψ and the discount rate α satisfies $\mu_c/(\alpha+\mu_c) > \psi$, then the α-discounted optimal threshold is finite and bounded as $\alpha \downarrow 0$.*

PROOF. Let g be modified unimodal and let α be close enough to 0 to satisfy $\mu_c/(\alpha+\mu_c) > \psi$. We have to prove that there exists a $k^* \in \mathbb{N}$ such that (4.39) is satisfied for all $k \geq k^*$. Since $\det(Y_i) > 0$, $i \in \mathbb{N}$, it is sufficient to establish this inequality componentwise, i.e.

$$g(k+1)K(k,i-1) \leq \frac{\mu_k}{\alpha+\mu_{k+1}} M(k,i)\, g(i), \qquad i = 2,\ldots,k, \tag{4.40a}$$

$$g(k+1)K(k,0)\det(B_1) \leq \alpha\lambda \frac{\mu_k}{\alpha+\mu_{k+1}} M(k,1)\, g(1). \tag{4.40b}$$

Let $i \geq c$. Note that by definition of $M(k,i)$ and $K(k,i)$, and $\mu_i = \mu_c$, the inequality (4.40a) is equivalent to

$$g(k+1) \leq \left(\frac{\mu_c}{\alpha+\mu_c}\right)^{k-i+1} g(i). \tag{4.41}$$

Clearly this is satisfied, since $\mu_c/(\alpha+\mu_c) > \psi$ and $g(k+1) \leq \psi^{k-i+1} g(i)$.

Let $i = 2, \ldots, c-1$. We shall show that there exists a k_i^* such that (4.40a) holds for $k \geq k_i^*$. It follows from the definition of K and M that (4.40a) is equivalent to

$$g(k+1) \prod_{j=i}^{k} \frac{\alpha+\mu_{j+1}}{\mu_j} \leq g(i). \tag{4.42}$$

Note that for $k \geq c$

$$0 \leq g(k+1) \prod_{j=i}^{k} \frac{\alpha+\mu_{j+1}}{\mu_j} \leq g(c) \left(\frac{\psi(\alpha+\mu_c)}{\mu_c}\right)^{k-c+1} \prod_{j=i}^{c-1} \frac{\alpha+\mu_{j+1}}{\mu_j}.$$

If we define k_i^* as

$$k_i^* = \min\left\{ m \in \mathbb{N} \;\middle|\; g(c) \left(\frac{\psi(\alpha+\mu_c)}{\mu_c}\right)^{k-c+1} \prod_{j=i}^{c-1} \frac{\alpha+\mu_{j+1}}{\mu_j} \leq g(i) \right\},$$

then k_i^* is finite, since $\prod_{j=i}^{c-1}$ is independent of m and $\psi(\alpha+\mu_c)/\mu_c < 1$. For all $k \geq k_i^*$ (4.42) is satisfied.

Finally consider the inequality (4.40b). Since $\det(B_1) = \alpha(\alpha+\lambda+\mu_1)$, this inequality is equivalent to

$$g(k+1)(\alpha+\lambda+\mu_1) \prod_{j=1}^{k} \frac{\alpha+\mu_{j+1}}{\mu_j} \leq \lambda g(1). \tag{4.43}$$

Analogously to the reasoning for $i = 2, \ldots, c-1$, there exists a k_1^* such that (4.43) is satisfied for $k \geq k_1^*$.

Taking $k^* = \max\{ k_i^* \mid i = 1,\ldots,c-1 \}$, gives (4.39) for all $k \geq k^*$. The fact that the α-discounted optimal threshold is bounded as $\alpha \downarrow 0$ is immediate from the observation that if the inequalities (4.42) and (4.43) hold for $\alpha_1 > 0$, then they also hold for all α_2, $0 \leq \alpha_2 \leq \alpha_1$. □

4.6 THE AVERAGE REWARD CONTROL PROBLEM

In this section we shall discuss the average reward control problem for the M/M/c queue with impatient customers. We shall show that we can get round the difficulties of proving the bounds of Theorems 3.3.8 or 3.3.9 by using the boundedness of the α-discounted optimal thresholds as $\alpha \downarrow 0$. We shall show that the average optimal policy is a threshold policy and that the design of the average optimal threshold becomes almost trivial.

Recall from Section 3.3.2.2 that the average reward control problem (P) was defined as

(P): maximise $W_u(n)$ over $u \in \mathcal{U}$ for all $n \in \mathbf{N}$.

Here the expected average reward $W_u : \mathbf{N} \to \mathbf{R}$ was defined as (recall (4.1))

$$W_u(n) = \liminf_{T\to\infty} \frac{W_u^T(n)}{T}, \qquad n \in \mathbf{N}. \tag{4.44}$$

The following method that describes how the average reward control problem (P) can be treated as the limit of the discounted reward control problem (P_α) as $\alpha \downarrow 0$, was inspired by the discrete time version in HEYMAN AND SOBEL [41, Section 4.6]. For this we need the following Tauberian theorem.

PROPOSITION 4.6.1 *Let $K : \mathbf{R}_+ \to \mathbf{R}_+$ be nondecreasing and let $f : \mathbf{R}_+ \to \mathbf{R}_+$,*

$$f(x) = \int_0^\infty e^{-xt}\, dK(t), \qquad x \in \mathbf{R}_+, \tag{4.45}$$

be well defined.

(i) $$\liminf_{t\to\infty} \frac{K(t)}{t} \le \liminf_{x\downarrow 0} x f(x) \le \limsup_{x\downarrow 0} x f(x) \le \limsup_{t\to\infty} \frac{K(t)}{t}.$$

(ii) If $\lim_{t\to\infty} K(t)/t$ exists, then $\lim_{x\downarrow 0} x f(x)$ exists and

$$\lim_{x\downarrow 0} x f(x) = \lim_{t\to\infty} \frac{K(t)}{t}.$$

PROOF. See WIDDER [108, p. 181–182]

Let k, $n_0 \in \mathbf{N}$. Denote the threshold policy with threshold k as u^k. If we define $K(t) = W_{u^k}^t(n_0)$, then clearly K satisfies the condition of Proposition 4.6.1, since $r(n,a)$ is bounded. For finite k the Markov chain $\mathcal{Y}$ under policy u^k is irreducible with the finite set $\{0,\ldots,k\}$ of ergodic states. Furthermore, the first passage time from any state to this set is almost surely finite and we may conclude that that under the policy u^k the average reward $\lim_{t\to\infty} K(t)/t$ exists and is independent of the initial state n_0 (cf. CHUNG [22, Section I.15]). As a result we have for any threshold policy u^k with finite k

$$\lim_{\alpha\downarrow 0} \alpha V_{u^k,\alpha} = \lim_{T\to\infty} \frac{W_{u^k}^T}{T} \tag{4.46}$$

since according to (4.2) taking $K(t) = W^t_{u^k}$ yields $f(\alpha) = V_{u^k,\alpha}$. If we denote for $\alpha \in (0,1)$ the α-optimal policy as u^α, then by the definition of optimality of policy u^α

$$\alpha V_{u,\alpha} \leq \alpha V_{u^\alpha,\alpha}, \qquad u \in \mathcal{U},\ \alpha > 0. \tag{4.47}$$

Let $(0,\delta)$ be a subset of $(0,1)$ such that for $0 < \alpha \leq \delta$ the optimal policy is a threshold policy denoted as u^{k_α}. Let $\{\alpha_n\}_{n=0}^\infty$ be a sequence in $(0,\delta)$ with $\alpha_n \downarrow 0$ as $n \to \infty$. Since k_{α_n} can take only finitely many values according to Lemma 4.5.4, there exists a subsequence, denoted as $\{\alpha_m\}_{m=0}^\infty$, with $k_{\alpha_m} = l$ constant. If we now take the limit along α_m, $m \to \infty$, in (4.46) and (4.47) we get for any policy $u \in \mathcal{U}$ and $n \in \mathbb{N}$

$$\liminf_{T\to\infty} \frac{W^T_u}{T} \leq \lim_{T\to\infty} \frac{W^T_{u^l}}{T} = W_{u^l},$$

so u^l is the policy that is optimal for the average reward control problem (P).

Once we know that the average optimal policy is a threshold policy with finite threshold, the design of the optimal threshold becomes almost trivial. Observe that the Markov chain $\mathcal{Y}$ with threshold k is an M/M/c/k queue, so the steady state probability $p_k(n)$ for state $n \in \mathbb{N}$ is (see Chapter 3)

$$p_k(n) = \frac{q(n)}{\sum_{i=0}^k q(i)}, \qquad n \in \mathbb{N},$$

where

$$q(n) = \prod_{j=1}^n \frac{\lambda}{\mu_j}, \qquad n \in \mathbb{N}.$$

Also observe that for threshold k the average return W_{u^k} is (cf. CHUNG [22])

$$W_{u^k} = \lambda g(n+1) \sum_{n=0}^{k-1} p_k(n).$$

It is a trivial exercise in arithmetic to show that $W_{u^k} \leq W_{u^{k+1}}$ is equivalent to

$$W_{u^k} \leq \frac{\lambda g(k+1) q(k)}{q(k+1)} = \mu_{k+1} g(k+1).$$

If we assume that the reward function g is modified unimodal, then this equivalence can be used to calculate the average optimal threshold with the following algorithm.

ALGORITHM 4.6.2

```
{ declarations }
HUGE: { arbitrary large integer };
k: integer;
numerator, denominator, q: real;
found: boolean;
{ initialisation }
k:=0;
found:=false;
q:=λ/μ1;
numerator:=0;
denominator:=q;
{ iteration }
while (k < HUGE) and not found
do
  if numerator/denominator > μk+1 g(k + 1)
  then found := true
  else
    k:=k+1;
    numerator:=numerator+q×λg(k);
    q:=q×λ/μk+1;
    denominator:=denominator+q;
  fi
od
```

4.7 NUMERICAL RESULTS

In this section we discuss some numerical examples of M/M/1 queues with impatient customers.

The first example deals with an M/M/1 queue with mean service time equal to 1.0. The deadlines of customers were generated from Erlang distributions of 1, 3 and 10 stages with mean 20.0. We have computed the optimal threshold for the average reward control problem. The values of the optimal threshold are depicted in Figure 4.2 for values of the arrival intensity λ ranging from 0.1 to 4.0. The plot suggests that for optimal operation of the queue the threshold should be varied with λ. For moderate to large values of λ, i.e. $\lambda > 0.5$, smaller thresholds are needed, if the variance of the deadline increases (compare for instance Erlang-1 with Erlang-10). This can be explained from the increasing probability of missing the deadline with the increasing variance. For small λ ($\lambda < 0.5$) we encounter the opposite observation. Although we have not been able to find an explanation for the phenomenon, one must realise that the difference in goodput for different values of the threshold in lightly loaded queues is neglectable.

We have investigated the robustness properties of threshold policies by computing the goodput (the average reward W_{u^k}) as a function of λ for several values of the threshold k and Erlang-3 distributed deadlines. The results of these computations are depicted in Figure 4.3. It appears that the threshold $k = 3$ that is optimal for

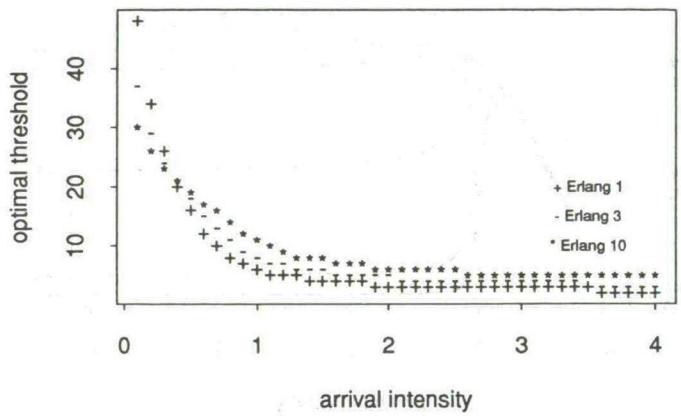

FIGURE 4.2. The optimal threshold in an M/M/1 queue.
Erlang distributed deadlines of mean 20 and 1, 3, and 10 stages.

large values of λ gives very poor goodput for λ in the range from 0.0 to 2.0. On the other hand, large values of k give the best performance for λ in the nominal operating region between 0.0 and 0.8. For large thresholds, however, the goodput decreases to 0 when λ becomes large. Our M/M/c queue model without admission control thus exhibits the same behaviour as an uncontrolled SPC switch under overload control (see Chapter 2). Note that for λ in the range 0.0 to 1.0 the goodputs for thresholds $k \geq 5$ differ only slightly (see also Table 4.1). From this we may conclude that the threshold policies are in fact quite robust, and it may for instance be satisfying to use a threshold $k = 6$ for all values of λ.

The second numerical example deals with the same example as before, but with a mean deadline of 50. Obviously thresholds can be taken larger than in the previous example, since customers are willing to wait longer for completion of their service (cf. Figure 4.4). The same remarks about robustness of the control can be made for this example (cf. Figure 4.5). For this example a threshold value of $k = 10$ seems an appropriate choice.

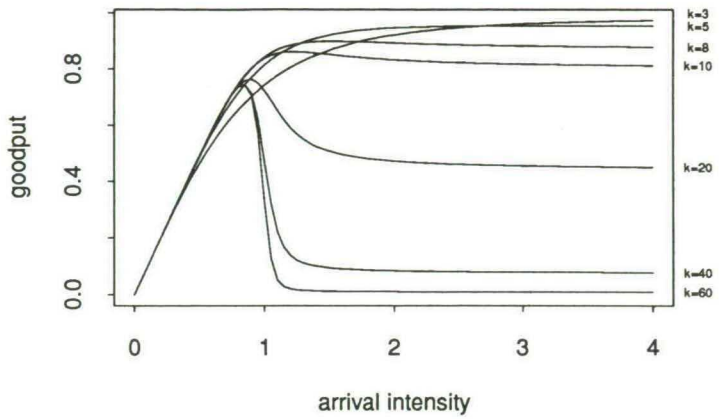

FIGURE 4.3. Goodput in an M/M/1 queue.
Erlang-3 distributed deadlines of mean 20.

k\λ	0.2	0.4	0.6	0.8	1.0	1.5	2.0	3.0	4.0
3	0.198	0.382	0.537	0.656	0.743	0.867	0.921	0.961	0.973
4	0.199	0.391	0.561	0.694	0.788	0.905	0.945	0.965	0.969
5	0.199	0.395	0.574	0.716	0.815	0.921	0.947	0.954	0.953
6	0.199	0.396	0.580	0.730	0.830	0.923	0.936	0.935	0.932
7	0.199	0.397	0.584	0.739	0.839	0.914	0.917	0.910	0.907
8	0.199	0.397	0.586	0.744	0.842	0.898	0.893	0.882	0.878
10	0.199	0.397	0.587	0.748	0.838	0.851	0.832	0.818	0.812
20	0.199	0.397	0.588	0.741	0.724	0.508	0.473	0.455	0.450
40	0.199	0.397	0.588	0.737	0.471	0.094	0.083	0.079	0.078
60	0.199	0.397	0.588	0.737	0.327	0.012	0.010	0.010	0.010

TABLE 4.1. GOODPUT IN AN M/M/1 QUEUE.
Erlang-3 distributed deadlines of mean 20.

k\λ	0.2	0.4	0.6	0.8	1.0	1.5	2.0	3.0	4.0
5	0.200	0.397	0.580	0.728	0.831	0.949	0.980	0.993	0.995
8	0.200	0.400	0.595	0.766	0.884	0.976	0.986	0.986	0.985
10	0.200	0.400	0.598	0.777	0.900	0.976	0.978	0.976	0.975
20	0.200	0.400	0.599	0.790	0.912	0.897	0.887	0.881	0.879
40	0.200	0.400	0.599	0.790	0.818	0.604	0.588	0.580	0.578
60	0.200	0.400	0.599	0.790	0.690	0.333	0.322	0.316	0.315
100	0.200	0.400	0.599	0.790	0.480	0.074	0.070	0.069	0.068
120	0.200	0.400	0.599	0.790	0.408	0.031	0.030	0.029	0.029

TABLE 4.2. GOODPUT IN AN M/M/1 QUEUE.
Erlang-3 distributed deadlines of mean 50.

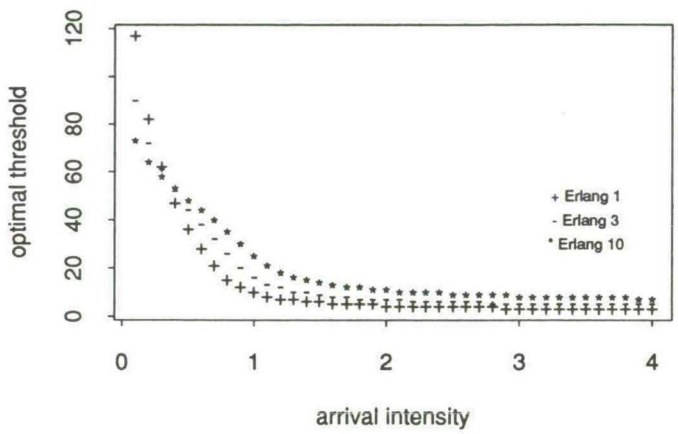

FIGURE 4.4. The optimal threshold in an M/M/1 queue. Erlang distributed deadlines of mean 50 and 1, 3, and 10 stages.

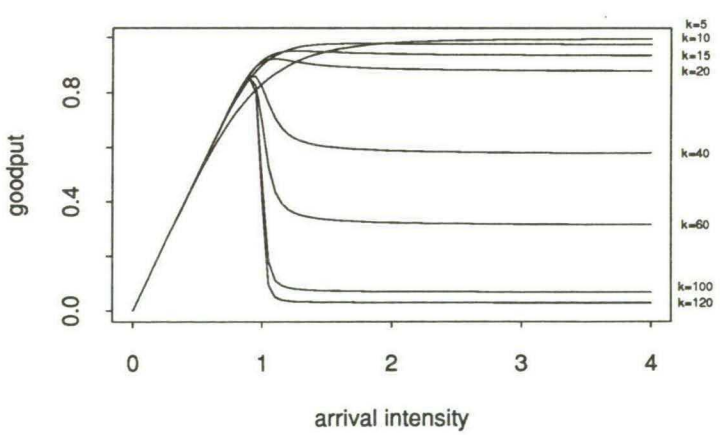

FIGURE 4.5. Goodput in an M/M/1 queue. Erlang-3 distributed deadlines of mean 50.

5

A Processor Sharing Queueing Model

5.1 Introduction

In this chapter we shall discuss a Processor Sharing queueing model for an SPC switch. This model is introduced to analyse the impact on the load of the processor caused by call requests and operator tasks. *Operator tasks* are issued by the operator and they are not directly related to call request processing. As examples one can think of administration and test procedures. As we have described in Chapter 2, the Central Control Unit (CCU) is a processor module running an operating system that is specially designed to maintain telephone traffic. Specifically this means that, apart from serving call requests, the CCU has to use part of its capacity to perform operator tasks. To guarantee proper operation of the switch, a part of the processor capacity has to be available to perform operator tasks at any time. What remains of the capacity may be used to handle call requests.

If the switch is only lightly loaded by call requests, then we can assign part of the remaining idle capacity to take care of operator tasks. If the switch is operating in an overload situation, that we assume is caused by call requests, then clearly all of the call handling capacity has to be assigned for this purpose. In such a situation operator tasks will receive only their guaranteed minimal portion of processor capacity.

In the model that we shall discuss the processor is represented by a Processor Sharing queue. This service discipline is an appropriate tool in the modelling of the time sharing operating system of a computer (cf. Kleinrock [52]). We consider an overload control algorithm that is allowed to reject or admit call requests and operator tasks. The information that will be available for the control algorithm consists of the number of call requests and the number of operator tasks that are in service. The main objective of the algorithm is to regulate the admission to the processor to maximise the number of completed call requests, under the constraints that the delay a request encounters in waiting and serving is limited and that a certain part of the processor's capacity is always available to deal with operator tasks. We can translate this objective into the terminology of queueing theory as: maximise the throughput of call requests

while maintaining an upper bound on their mean sojourn times and at the same time guaranteeing a lower bound on the throughput of operator tasks. In this formulation the impatience of subscribers is thus implicitly accounted for by the constraint on the mean sojourn times of call requests. We have chosen this approach to model impatience, since explicit modelling would require the derivation of the sojourn time distribution of call requests in a Processor Sharing queue which also handles operator tasks. Although quite recently progress has been made in this area (cf. VAN DEN BERG [8]), it is not yet clear how these results can be put to advantage in optimal control problems. Alternative approaches to implicitly account for impatience shall be mentioned in the references later in this section.

In this chapter we shall restrict our attention to two classes of admission policies. In both classes the decision to admit new call requests and operator tasks is non-randomised, i.e. requests and tasks are either rejected or admitted. If a task or request is admitted, it joins the queue for service, otherwise it is lost. The first class of admission policies, called *partitioning policies*, is implemented by two separate thresholds, one for call requests and one for operator tasks. If the number of call requests in service reaches the corresponding threshold, a new request is rejected until the number of requests in service decreases by a service completion. An analogous algorithm is employed for operator tasks. The second class of admission policies that we consider, are called *sharing policies*. A sharing policy uses a pool that contains c permission units (*p.u.'s*) for some $c \in \mathbb{N}$. If a call request arrives at the queue, it can be admitted only if at least c_1, $c_1 \in \mathbb{N}$, of p.u.'s are available in the pool. If these p.u.'s are available, then they are removed from the pool and the request is allowed to enter the queue (cf. Figure 5.1). After the call request has completed its service, the p.u.'s of this request are put back into the pool. An analogous strategy is used for operator tasks, though c_2, the number of p.u.'s needed for admission of a task, may differ from c_1. This control was implemented to allow "sharing" of the processor capacity, i.e. when the demand for service of call requests is low, then operator tasks are allowed to use part of the call handling capacity and vice versa.

In this chapter we shall discuss the constrained control problem, that was introduced in the beginning of this section, for both classes of admission policies. We prove that the problem can be solved for partitioning policies. Although we have not been able to solve the problem for sharing policies, we shall discuss the performance of such policies for several parameter settings.

Constrained optimisation problems for queues have gained increasing interest of researchers during the last five years. In ROSS [75] finite state Markov Decision Processes are considered with a reward and cost structure. The objective is to maximise the average return with a constraint on the average costs. In HORDIJK AND SPIEKSMA [45] an optimisation problem with a constraint on the average cost for a one-dimensional queueing system is discussed. They show that for rather general assumptions on the cost and reward structure the optimal control randomises in exactly one state. In MA AND MAKOWSKI [63] a constrained optimal flow control problem is solved with Lagrangian methods. Again the optimal control is shown to be randomised in exactly one state. A well known reference on constrained optimal flow control is ROBERTAZZI AND LAZAR [74]. They consider the problem of maximising the throughput of a one-dimensional queue under a constraint on the average delay. The optimal control is shown to be of a window flow type and again it is randomised in exactly one state. In

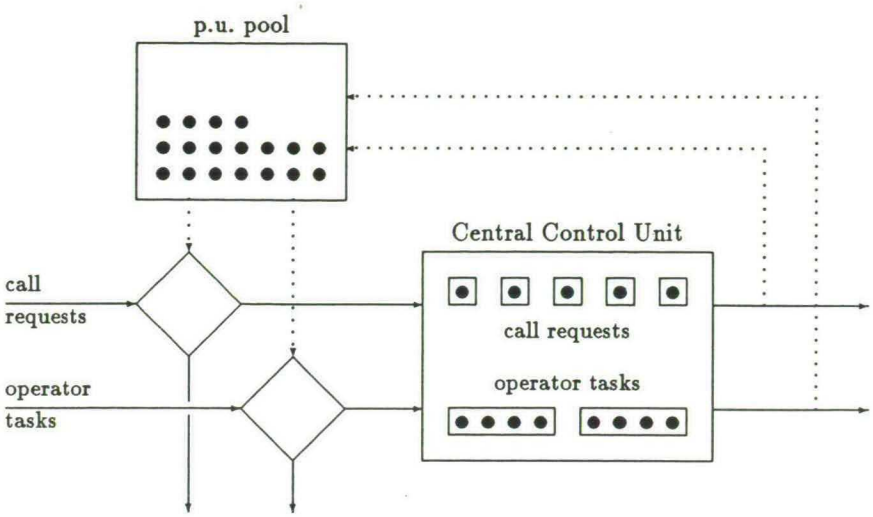

FIGURE 5.1. A Processor Sharing queue with a pool of permission units.

NAIN AND ROSS [65] a queueing model is discussed for multiplexing heterogeneous Poisson arrival streams onto a single communication channel. The optimisation problem is to minimise a linear combination of the average delays, while at the same time subjecting the average delay for one stream to a hard constraint. This constraint is an alternative approach to account for impatience of customers implicitly. The optimal multiplexing policy is shown to be a randomised modification of the μc-rule. In NAIN AND ROSS [66] a similar system is considered with renewal arrival streams. To minimise a linear combination of the average queue lengths with a hard constraint on the average queue length of one arrival stream, the optimal multiplexing policy is shown to be a randomised modification of a static-priority rule. The performance analysis of partitioning and sharing policies is related to the analysis of resource sharing in KAUFMAN [49]. He derives a product form for the equilibrium distribution of a queueing model with sharing policies and presents a one-dimensional recursion for the computation of performance measures. In FOSCHINI AND GOPINATH [37] an optimal control problem for a queueing system with two processors and a common waiting room is presented. They show that the control that minimises a weighted sum of the idle times of the servers, is a combination of a partitioning and a sharing policy.

This chapter is organised as follows. In Section 5.2 we present a description of the queueing model and formulate the control problems for the partitioning and sharing policies. Partitioning policies are discussed in Section 5.3. In Section 5.3.1 we derive monotonicity properties of performance measures, that are used in Section 5.3.2 to establish conditions for the existence of an optimal policy. A design algorithm for the optimal policy is discussed in Section 5.3.3. We address the performance of sharing policies in Section 5.4. The chapter is concluded with some numerical examples in Section 5.5.

5.2 Description of the queueing model

Consider a Processor Sharing queue with two independent Poisson arrival processes. Customers of class 1 represent the call requests and they arrive with rate λ_1. Operator tasks are referred to as class 2 customers and they arrive with rate λ_2 (see Figure 5.2).

The service requirements for customers of class i, are independent and exponentially distributed with mean $1/\mu_i$, $i = 1, 2$. We define the workloads $\rho_i = \lambda_i/\mu_i$, $i = 1, 2$. The server works with constant service rate according to the Processor Sharing discipline, i.e. at any time each customer receives the same amount of service as any other customer. In Kelly's terminology (see page 19) this means that

$$\phi(k,i) = \frac{1}{k}, \qquad k \in \mathbb{N},\ i = 0, \ldots, k,$$

$$\delta(k,i) = \frac{1}{k+1}, \qquad k \in \mathbb{N},\ i = 0, \ldots, k+1.$$

Processor Sharing is thus a symmetric service discipline (cf. Definition 3.2.1).

The queueing process is a continuous time Markov process $\mathcal{X} = (X_t, t \geq 0)$ on the state space $\mathcal{S} = \mathbb{N}^2$, where a state is represented by a population vector $\mathbf{m} = (m_1, m_2)$ corresponding to a population of m_i customers of class i, $i = 1, 2$. This means that we do not distinguish different customers within one class, but only keep track of the population of both classes.

Upon arrival to the queue customers can either be admitted or rejected according to a stationary admission policy. In accordance with the notation in Section 3.2 we can represent such a stationary admission policy by two arrival probability functions A_1, $A_2 : \mathbb{N}^2 \to \{0, 1\}$. If the queue population is $\mathbf{m} \in \mathbb{N}^2$, then a new arriving customer of class i is rejected or admitted if $A_i(\mathbf{m}) = 0$ or 1, respectively. Admitted customers join the queue until completion of service, while rejected customers are assumed lost without further influencing the future of the arrival and queueing processes.

In the remaining sections of this chapter we shall restrict our attention to admission policies that are described by coordinate convex subsets of $\mathbb{N}^2$ (cf. Example 3.2.7 on page 24). If we let $\mathcal{C} \subset \mathbb{N}^2$ denote a coordinate convex set, then we define the stationary policy $f_{\mathcal{C}}$ as the admission policy that restricts the state space to $\mathcal{C}$. In words this means that customers are admitted only if this does not result in a population vector outside of the set $\mathcal{C}$, so

$$A^i(\mathbf{m}) = 1(\mathbf{m} + \mathbf{e}_i \in \mathcal{C}), \qquad \mathbf{m} \in \mathbb{N}^2,\ i = 1, 2. \tag{5.1}$$

The set $\mathcal{C}$ is called the admission region of $f_{\mathcal{C}}$. We do not exercise any control on departures of customers. In this chapter we shall make a restriction to two subclasses of coordinate convex sets in $\mathbb{N}^2$.

Definition 5.2.1 (Partitioning Policies)
If for some M_1, $M_2 \in \mathbb{N} \cup \{\infty\}$

$$\mathcal{C} = \{\, \mathbf{m} \in \mathbb{N}^2 \mid 0 \leq m_1 \leq M_1, 0 \leq m_2 \leq M_2 \,\}, \tag{5.2}$$

then $f_{\mathcal{C}}$ *is called a* partitioning policy with thresholds M_i for class i, $i = 1, 2$. *The set* $\mathcal{C}$ *is called a* partition *and the set of partitioning policies is denoted as* $\mathcal{F}_P$.

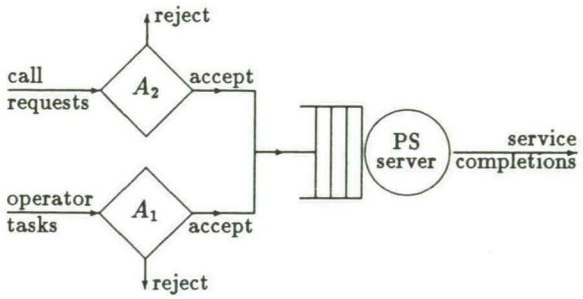

FIGURE 5.2. A Processor Sharing queueing model.

The state space of the queueing process $\mathcal{X}$ restricted to a partition with thresholds $M_1 = 6$ and $M_2 = 4$ is depicted in Figure 5.3. The effect of such a policy is that the separate admission policies for both classes become "independent", i.e. the population of one class does not affect the admission probability for the other class. One can also say that the admission policy for one class uses only partial state information for its decision, viz. it uses only the population size of its corresponding class.

The second class of coordinate convex admission regions corresponds to triangles in $\mathbb{N}^2$.

DEFINITION 5.2.2 (SHARING POLICIES)
If for some $c_1, c_2 \in \mathbb{N}$, $c \in \mathbb{N} \cup \{\infty\}$

$$\mathcal{C} = \{ \mathbf{m} \in \mathbb{N}^2 \mid 0 \leq c_1 m_1 + c_2 m_2 \leq c \}, \tag{5.3}$$

then $f_{\mathcal{C}}$ *is called a* sharing policy *with* weight factors c_i *for class* i, $i = 1, 2$, *and* poolsize c. *The set of sharing policies is denoted as* $\mathcal{F}_S$.

An example of the state space of a sharing policy with weight factors $c_1 = 1$, $c_2 = 2$ and poolsize $c = 8$ is depicted in Figure 5.4.

The terms partitioning and sharing were used in KAUFMAN [49] in the context of memory allocation policies for computers (see also FOSCHINI AND GOPINATH [37]).

Since both the partitioning and the sharing policies allow infinite state spaces, we need some conditions that guarantee that the controlled queueing process is ergodic. It is easy to check that for ergodicity the following conditions are sufficient.

CONDITION 5.2.3 (ERGODICITY CONDITIONS)
Let $\mathcal{C}$ *be the admission region of policy* $f_{\mathcal{C}}$. *The queueing process that is restricted to the set* $\mathcal{C}$ *is ergodic if one of the following conditions is true.*

(i) If $\limsup_{\mathbf{m} \in \mathcal{C}} m_1 = \infty$, *then* $\rho_1 < 1$.

(ii) If $\limsup_{\mathbf{m} \in \mathcal{C}} m_2 = \infty$, *then* $\rho_2 < 1$.

(iii) If $\limsup_{\mathbf{m} \in \mathcal{C}} m_1 = \limsup_{\mathbf{m} \in \mathcal{C}} m_2 = \infty$, *then* $\rho_1 + \rho_2 < 1$.

(iv) If $\mathcal{C}$ *is a finite set, then the queueing process is ergodic for all values of* λ_1, λ_2, μ_1, μ_2.

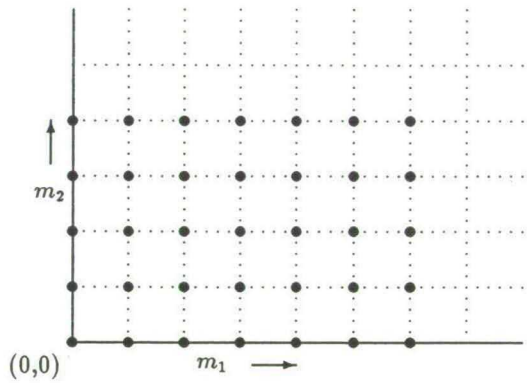

FIGURE 5.3. State space for a partitioning policy.
Thresholds $M_1 = 6$ and $M_2 = 4$.

We can now proceed with the formulation of the performance measures throughput, expected sojourn time and expected queue length. Let $\mathcal{C}$ be a coordinate convex subset of $\mathbb{N}^2$. We introduce for $i = 1, 2$, $n \in \mathbb{N}$, $X_0 \in \mathbb{N}^2$, $t \geq 0$ and $h > 0$ the random variables

$\mathbf{S}_i(n, \mathcal{C}, X_0)$ The sojourn time or residence time in the queue of the n-th admitted customer of class i under admission policy $f_{\mathcal{C}}$ when the initial state is X_0.

$\mathbf{L}_i(t, \mathcal{C}, X_0)$ The number of customers of class i in the queue at time t under policy $f_{\mathcal{C}}$ when the initial state is X_0.

$\mathbf{T}_i(t, t+h, \mathcal{C}, X_0)$ The number of customers of class i that depart from the queue in the interval $[t, t+h)$ under policy $f_{\mathcal{C}}$ when the initial state is X_0.

If we assume that the ergodicity conditions are satisfied, then we can define

$$S_i(\mathcal{C}) = \lim_{n \to \infty} E\big[\mathbf{S}_i(n, \mathcal{C}, X_0)\big],$$

$$L_i(\mathcal{C}) = \lim_{t \to \infty} E\big[\mathbf{L}_i(t, \mathcal{C}, X_0)\big],$$

$$T_i(\mathcal{C}) = \lim_{h \downarrow 0} \lim_{t \to \infty} \frac{E\big[\mathbf{T}_i(t, t+h, \mathcal{C}, X_0)\big]}{h}.$$

These quantities are well defined and independent of the initial state X_0. It is well known from literature that these three performance measures can be expressed in the equilibrium distribution $\pi_{\mathcal{C}}$ of process $\mathcal{X}_{\mathcal{C}}$ (cf. REISER AND KOBAYASHI [72]). From Theorem 3.2.10 we know that the equilibrium distribution $\pi_{\mathcal{C}}(\mathbf{m})$ of state $\mathbf{m} = (m_1, m_2)$ is

$$\pi_{\mathcal{C}}(\mathbf{m}) = G(\mathcal{C})\, 1(\mathbf{m} \in \mathcal{C}) \binom{m_1 + m_2}{m_1} \rho_1^{m_1} \rho_2^{m_2}, \qquad \mathbf{m} \in \mathbb{N}^2, \qquad (5.4)$$

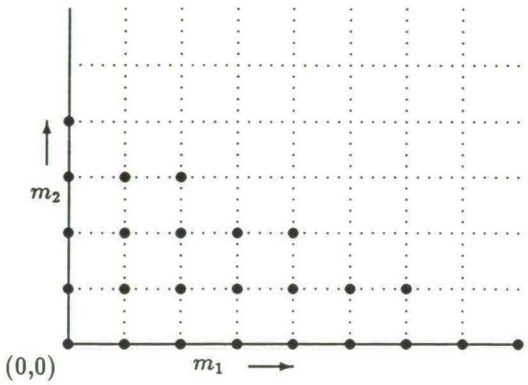

FIGURE 5.4. State space for a sharing policy.
Weight factors $c_1 = 1$, $c_2 = 2$ and poolsize $c = 8$.

where $G(\mathcal{C})$ is the normalisation constant that guarantees that $\sum_{\mathbf{m}\in\mathcal{C}} \pi_{\mathcal{C}}(\mathbf{m}) = 1$. The expressions for the *throughput* $T_i(\mathcal{C})$ and the *expected queue length* $L_i(\mathcal{C})$ for class i under admission policy $f_{\mathcal{C}}$ are

$$T_i(\mathcal{C}) = \sum_{\mathbf{m}\in\mathcal{C}} \pi_{\mathcal{C}}(\mathbf{m})\,\lambda_i\,1(\mathbf{m}+\mathbf{e}_i \in \mathcal{C}), \qquad i = 1,2, \tag{5.5}$$

$$= \sum_{\mathbf{m}\in\mathcal{C}} \pi_{\mathcal{C}}(\mathbf{m})\frac{m_i\mu_i}{m_1+m_2}1(m_1+m_2>0), \qquad i = 1,2, \tag{5.6}$$

$$L_i(\mathcal{C}) = \sum_{\mathbf{m}\in\mathcal{C}} \pi_{\mathcal{C}}(\mathbf{m})m_i, \qquad i = 1,2. \tag{5.7}$$

The *expected sojourn time* S_i *for class* i can be expressed in T_i and L_i through Little's law (cf. LITTLE [62]):

$$L_i(\mathcal{C}) = T_i(\mathcal{C})S_i(\mathcal{C}), \qquad i = 1,2. \tag{5.8}$$

With these definitions we are now all set to introduce the constrained control problems.

DEFINITION 5.2.4 *Let* S *and* T *be fixed constants in* $\mathbf{R}_+$ *satisfying* $S \geq 1/\mu_1$ *and* $T \leq \lambda_2$. *The problems* (P_P) *and* (P_S) *are defined as*

(P_P): *maximise* $T_1(\mathcal{C})$ *over* $f_{\mathcal{C}} \in \mathcal{F}_P$ *under the constraints* $S_1(\mathcal{C}) \leq S$ *and* $T_2(\mathcal{C}) \geq T$. *A partition* $\mathcal{C}$ *is called* optimal *for* (P_P) *if it satisfies the constraints on* S_1 *and* T_2 *and attains the maximum for* T_1.

(P_S): *maximise* $T_1(\mathcal{C})$ *over* $f_{\mathcal{C}} \in \mathcal{F}_S$ *under the constraints* $S_1(\mathcal{C}) \leq S$ *and* $T_2(\mathcal{C}) \geq T$. *A sharing policy* $f_{\mathcal{C}}$ *is called* optimal *for* (P_S) *if it satisfies the constraints on* S_1 *and* T_2 *and attains the maximum for* T_1.

The condition $S \geq 1/\mu_1$ is to guarantee that the constrained problems are not trivial, since the mean sojourn time of a class 1 customer can never be smaller than its mean service time. For the same reason the condition $T \leq \lambda_2$ was introduced, since the throughput of class 2 customers can never be larger than the arrival rate.

We shall call a policy $f_C \in \mathcal{F}_P$ *feasible for* (P_P) if the queueing process $\mathcal{X}$ is ergodic under f_C and $S_1(\mathcal{C}) \leq S$ and $T_2(\mathcal{C}) \geq T$. An analogous definition for feasibility of sharing policies is used.

We conclude this section with a remark about the choice of the exponential service time distributions. As we have seen in Section 3.2 the product form equilibrium distribution remains valid even for general service time distributions. It is also known (cf. for instance KELLY [50] and REISER AND KOBAYASHI [72]) that the throughput, expected queue length and sojourn time depend only on the mean of the service time distributions. As a consequence the problems (P_P) and (P_S) are in fact defined for general service time distributions, and the optimal policy depends only on the mean of the service times. A preliminary result is thus the robustness of the optimal admission policies with respect to the service time distributions. In an attempt to keep the notation simple, however, we shall restrict ourselves to the exponential case in the remainder of this chapter.

We shall address the constrained optimisation problems (P_P) and (P_S) in Sections 5.3 and 5.4, respectively.

5.3 PARTITIONING POLICIES

5.3.1 Expressions for performance measures

In this section we shall discuss the constrained control problem (P_P) for partitioning policies. We shall derive expressions for the performance measures T_1, T_2, and S_1 as functions of a partition's thresholds. We shall also establish monotonicity results for these performance measures with respect to the thresholds. These results are employed to formulate existence conditions for an optimal policy in Section 5.3.2 and to design a search procedure for the optimal thresholds in Section 5.3.3.

First we introduce some notation. For a partitioning policy $f_C \in \mathcal{F}_P$ with thresholds M_1 and M_2 we denote $\mathcal{C}$ as $\langle M_1, M_2 \rangle$, $G(\mathcal{C}) = G(M_1, M_2)$, and for $i = 1, 2$, $T_i(\mathcal{C}) = T_i(M_1, M_2)$, $L_i(\mathcal{C}) = L_i(M_1, M_2)$ and $S_i(\mathcal{C}) = S_i(M_1, M_2)$. We allow M_1 and M_2 to be infinite. We start with the derivation of expressions for the performance measures. For this it is convenient to introduce the unnormalised equilibrium distribution $\bar{\pi} : \mathbf{N}^2 \to \mathbf{R}_+$,

$$\bar{\pi}(\mathbf{m}) = \binom{m_1 + m_2}{m_1} \rho_1^{m_1} \rho_2^{m_2}, \qquad \mathbf{m} \in \mathbf{N}^2. \tag{5.9}$$

From (5.4) the normalising constant $G(\mathcal{C})$ for a partition $\mathcal{C} = \langle M_1, M_2 \rangle$, can be expressed as

$$G(M_1, M_2) = \left[\sum_{m_1=0}^{M_1} \sum_{m_2=0}^{M_2} \bar{\pi}(\mathbf{m}) \right]^{-1}. \tag{5.10}$$

Note that $G(\mathcal{C})$ is defined only if the ergodicity conditions are satisfied, i.e. $\rho_1 < 1$ if $M_1 = \infty$, $\rho_2 < 1$ if $M_2 = \infty$ and $\rho_1 + \rho_2 < 1$ if $M_1 = M_2 = \infty$. The performance measures can now be expressed as

$$T_1(M_1, M_2) = \lambda_1 G(M_1, M_2) \sum_{m_1=0}^{M_1-1} \sum_{m_2=0}^{M_2} \bar{\pi}(m_1, m_2), \tag{5.11}$$

$$T_2(M_1, M_2) = \lambda_2 G(M_1, M_2) \sum_{m_1=0}^{M_1} \sum_{m_2=0}^{M_2-1} \bar{\pi}(m_1, m_2), \tag{5.12}$$

$$L_i(M_1, M_2) = G(M_1, M_2) \sum_{m_1=0}^{M_1} \sum_{m_2=0}^{M_2} m_i \, \bar{\pi}(m_1, m_2), \qquad i = 1, 2. \tag{5.13}$$

The expected sojourn time follows from Little's law (5.8):

$$S_i(M_1, M_2) = \frac{L_i(M_1, M_2)}{T_i(M_1, M_2)}, \qquad i = 1, 2. \tag{5.14}$$

Again the remark about the ergodicity conditions applies. For finite thresholds the performance measures can be computed quite easily by a recursive scheme in M_1 and M_2 (cf. REISER AND KOBAYASHI [72]). However, since a feasible partition may have one or two infinite thresholds, we must also be able to compute the measures for these cases. To accomplish this we define for $n \in \mathbb{N}$, $0 \leq x < 1$, $y \geq 0$:

$$J_1(n, x, y) = \sum_{j=0}^{n} \sum_{k=0}^{j} \binom{j}{k} \frac{y^j \, x^{j-k}}{(1-x)^{j+1-k}}, \tag{5.15}$$

$$J_2(n, x, y) = \sum_{j=0}^{n} \sum_{k=0}^{j+1} \binom{j+1}{k} \frac{(j+1)\, y^j \, x^{j+2-k}}{(1-x)^{j+2-k}}, \tag{5.16}$$

$$J_3(n, x, y) = \sum_{j=1}^{n} \sum_{k=0}^{j} \binom{j}{k} \frac{j \, y^j \, x^{j+1-k}}{(1-x)^{j+1-k}}. \tag{5.17}$$

The computation of these values can be implemented into an algorithm by recursion in n. Furthermore, the performance measures with infinite thresholds can be expressed in the finite sums J_1, J_2, J_3, as is illustrated in the following lemma.

LEMMA 5.3.1 *For all finite* $M_1, M_2 \in \mathbb{N}$

$$T_1(\infty, M_2) = \lambda_1 = \lim_{M_1 \to \infty} T_1(M_1, M_2), \quad \rho_1 < 1,$$

$$S_1(\infty, M_2) = \frac{J_2(M_2, \rho_1, \rho_2)}{\lambda_1 J_1(M_2, \rho_1, \rho_2)} = \lim_{M_1 \to \infty} S_1(M_1, M_2), \quad \rho_1 < 1,$$

$$T_2(\infty, M_2) = \lambda_2 \frac{J_1(M_2 - 1, \rho_1, \rho_2)}{J_1(M_2, \rho_1, \rho_2)} = \lim_{M_1 \to \infty} T_2(M_1, M_2), \quad \rho_1 < 1,$$

$$T_1(M_1, \infty) = \lambda_1 \frac{J_1(M_1 - 1, \rho_2, \rho_1)}{J_1(M_1, \rho_2, \rho_1)} = \lim_{M_2 \to \infty} T_1(M_1, M_2), \quad \rho_2 < 1,$$

$$S_1(M_1, \infty) = \frac{J_3(M_1, \rho_2, \rho_1)}{\lambda_1 J_1(M_1 - 1, \rho_2, \rho_1)} = \lim_{M_2 \to \infty} S_1(M_1, M_2), \quad \rho_2 < 1,$$

$$T_2(M_1, \infty) = \lambda_2 = \lim_{M_2 \to \infty} T_2(M_1, M_2), \quad \rho_2 < 1.$$

PROOF. Examination of expressions (5.11)–(5.14) learns that we have to prove for $0 \leq x < 1$, $y \geq 0$,

$$J_1(n,x,y) = \lim_{k\to\infty} \sum_{i=0}^{k} \sum_{j=0}^{n} \binom{i+j}{i} x^i y^j, \tag{5.18}$$

$$J_2(n,x,y) = \lim_{k\to\infty} \sum_{i=0}^{k} \sum_{j=0}^{n} \binom{i+j}{i} i\, x^i y^j, \tag{5.19}$$

$$J_3(n,x,y) = \lim_{k\to\infty} \sum_{i=0}^{k} \sum_{j=0}^{n} \binom{i+j}{i} j\, x^i y^j. \tag{5.20}$$

Let $0 \leq x < 1$ and $y \geq 0$. We can rewrite (5.18) as

$$\lim_{k\to\infty} \sum_{j=0}^{n} \frac{y^j}{j!} \sum_{i=0}^{k} \frac{(i+j)!}{i!} x^i$$

$$= \lim_{k\to\infty} \sum_{j=0}^{n} \frac{y^j}{j!} \frac{d^j}{dx^j} \sum_{i=0}^{k} x^{i+j}$$

$$= \sum_{j=0}^{n} \frac{y^j}{j!} \frac{d^j}{dx^j} \sum_{i=0}^{\infty} x^{i+j}$$

$$= \sum_{j=0}^{n} \frac{y^j}{j!} \frac{d^j}{dx^j} \left[\frac{x^j}{1-x}\right]$$

$$= \sum_{j=0}^{n} \frac{y^j}{j!} \sum_{k=0}^{j} \binom{j}{k} \left[\frac{d^k}{dx^k} x^j\right] \left[\frac{d^{j-k}}{dx^{j-k}} \frac{1}{1-x}\right]$$

$$= J_1(n,x,y).$$

The interchange of the differential operator and the infinite sum is allowed since $x < 1$. Analogously we get for $0 \leq x < 1$, $y \geq 0$ from (5.19)

$$\lim_{k\to\infty} \sum_{j=0}^{n} \frac{y^j}{j!} \sum_{i=1}^{k} \frac{(i+j)!}{(i-1)!} x^i$$

$$= \sum_{j=0}^{n} \frac{x\, y^j}{j!} \frac{d^{j+1}}{dx^{j+1}} \left[\frac{x^{j+1}}{1-x}\right]$$

$$= J_2(n,x,y).$$

Finally, for $0 \leq x < 1$, $y \geq 0$, we can derive J_3 in a similar way as (5.18), since it equals

$$\sum_{j=1}^{n} \frac{y^j}{(j-1)!} \sum_{i=0}^{\infty} \frac{(i+j)!}{i!} x^i.$$

□

For the performance measures we can derive some intuitively appealing inequalities as is shown by the following lemma.

LEMMA 5.3.2 (MONOTONICITY PROPERTIES)
For all M_1, $M_2 \in \mathbb{N}$ *and* $i = 1, 2$,

$$T_1(M_1, M_2) < T_1(M_1 + 1, M_2), \quad T_1(M_1, M_2) > T_1(M_1, M_2 + 1), \tag{5.21}$$

$$T_2(M_1, M_2) > T_2(M_1 + 1, M_2), \quad T_2(M_1, M_2) < T_2(M_1, M_2 + 1), \tag{5.22}$$

$$L_i(M_1, M_2) < L_i(M_1 + 1, M_2), \quad L_i(M_1, M_2) < L_i(M_1, M_2 + 1), \tag{5.23}$$

$$S_i(M_1, M_2) < S_i(M_1 + 1, M_2), \quad S_i(M_1, M_2) < S_i(M_1, M_2 + 1). \tag{5.24}$$

PROOF. See DE WAAL AND VAN DIJK [102].

Note that these inequalities are indeed what we would intuitively expect. For example (5.21) states that increasing the threshold M_1 for class 1 while keeping M_2 constant, gives a higher throughput T_1, since obviously more class 1 customers are going to be admitted to the queue. Conversely the throughput T_2 for class 2 will decrease as less processor capacity will be left to serve class 2 customers.

In DE WAAL AND VAN DIJK [102] the monotonicity properties of Lemma 5.3.2 are proven for a service discipline that is more general than standard Processor Sharing. This generalisation includes the model in COHEN [23], and also non-symmetric modifications. The monotonicity of throughputs and mean queue lengths is proven by treating these performance measures as average rewards and showing that monotonicity exists for finite horizon rewards. The inequality (5.24) for S_2 is an immediate result of (5.22) and (5.23) by application of Little's law (5.8). The inequality (5.24) for S_1 can, however, not be derived in this manner and has to be proven explicitly. This is the only monotonicity property where the product form equilibrium distribution is actually needed.

The monotonicity results of Lemma 5.3.2 remain valid when one of the thresholds becomes infinite, though for obvious reasons the inequality signs $<$ and $>$ have to be replaced by $\leq$ and $\geq$, respectively.

5.3.2 Existence of an optimal partition

We shall now use the monotonicity results of the preceding section to give conditions for the existence of a solution to problem (P_P). The first lemma shows that the existence of a feasible partition is sufficient for the existence of an optimal partition. This result is not trivial, since the existence of a feasible partition does not guarantee that there is a partition that attains the maximum class 1 throughput.

LEMMA 5.3.3 *There exists a feasible policy for problem* (P_P), *if and only if there exists an optimal policy.*

PROOF. Since an optimal partition is by definition feasible, the "only if" statement is immediate. The proof for the "if" part proceeds by contradiction. Assume that there is a feasible partition, but no optimal partition. Since T_1 is bounded by λ_1, this means that there is no feasible partition that attains a maximum class 1 throughput and thus there must be infinitely many feasible partitions. Consequently there exists a sequence $\{\mathcal{C}_n\}_{n=0}^{\infty}$ of feasible partitions satisfying

(a.i) $T_1(\mathcal{C}_n)$ is strictly increasing for $n \in \mathbb{N}$.

(a.ii) For any feasible partition $\mathcal{C}$ there exists an $n_{\mathcal{C}} \in \mathbb{N}$ such that $T_1(\mathcal{C}_n) > T_1(\mathcal{C})$ for $n > n_{\mathcal{C}}$.

In the remainder of this proof we shall use the notation $M_i(\mathcal{C})$ for $i = 1, 2$, to denote the class i threshold of partition $\mathcal{C}$. Note that $M_i(\mathcal{C})$ can be infinite.

First we shall prove that

$$\sup_{n \in \mathbb{N}} M_1(\mathcal{C}_n) = \infty. \tag{5.25}$$

Assume that (5.25) is not true, so $M_1(\mathcal{C}_n)$ takes only a finite number of discrete values. Consequently there must exist a subsequence $\{\mathcal{C}_n^{(1)}\}_{n=0}^{\infty}$ of $\{\mathcal{C}_n\}_{n=0}^{\infty}$, with $M_1(\mathcal{C}_n^{(1)})$ constant and equal to $\limsup_{n\to\infty} M_1(\mathcal{C}_n)$ which we shall denote as $\overline{M}_1$. Since $T_1(\mathcal{C}_n)$ and thus also $T_1(\mathcal{C}_n^{(1)})$ is strictly increasing, $M_2(\mathcal{C}_n^{(1)})$ must take infinitely many different values due to (5.21). Therefore, we can construct a subsequence $\{\mathcal{C}_n^{(2)}\}_{n=0}^{\infty}$ of $\{\mathcal{C}_n^{(1)}\}_{n=0}^{\infty}$ such that

(b.i) $M_1(\mathcal{C}_n^{(2)})$ is constant (by definition of $\mathcal{C}_n^{(1)}$ and thus also of $\mathcal{C}_n^{(2)}$).

(b.ii) $M_2(\mathcal{C}_n^{(2)})$ is strictly increasing in $n \in \mathbb{N}$.

(b.iii) $T_1(\mathcal{C}_n^{(2)})$ is strictly increasing in $n \in \mathbb{N}$.

This is in contradiction with the monotonicity property (5.21) of Lemma 5.3.2, however. We may thus conclude that $\sup_{n\in\mathbb{N}} M_1(\mathcal{C}_n) = \infty$.

If we have a feasible partition $\mathcal{C}_n$ for some $n \in \mathbb{N}$ with $M_1(\mathcal{C}_n) = \infty$, then $T_1(\mathcal{C}_n) = \lambda_1$ by Lemma 5.3.1, and this contradicts (a.ii). We may therefore assume that for all $n \in \mathbb{N}$, $M_1(\mathcal{C}_n) < \infty$. Since $\sup_{n\in\mathbb{N}} M_1(\mathcal{C}_n) = \infty$, there must exist a subsequence $\{\mathcal{C}_n^{(3)}\}_{n=0}^{\infty}$ of $\{\mathcal{C}_n\}_{n=0}^{\infty}$ with $M_1(\mathcal{C}_n^{(3)}) \uparrow \infty$ as $n \to \infty$. With regard to the sequence $\{M_2(\mathcal{C}_n^{(3)})\}_{n=0}^{\infty}$ we can distinguish between the following two cases:

(c.i) $M_2(\mathcal{C}_n^{(3)})$ takes only finitely many values for $n \in \mathbb{N}$.

(c.ii) $M_2(\mathcal{C}_n^{(3)})$ takes infinitely many values for $n \in \mathbb{N}$.

If (c.i) holds, then we can construct (yet another) subsequence $\{\mathcal{C}_n^{(4)}\}_{n=0}^{\infty}$ of $\{\mathcal{C}_n^{(3)}\}_{n=0}^{\infty}$, such that $M_2(\mathcal{C}_n^{(4)})$ is constant and equal to $\limsup_{n\to\infty} M_2(\mathcal{C}_n^{(3)})$ which we shall denote as $\overline{M}_2$. For all $n \in \mathbb{N}$ we then have by the definition of feasibility, $S_1(M_1(\mathcal{C}_n^{(4)}), \overline{M}_2)$ increasing and bounded from above by S, and $T_2(M_1(\mathcal{C}_n^{(4)}), \overline{M}_2)$ decreasing and bounded from below by T. Furthermore, by definition of $\{\mathcal{C}_n^{(3)}\}_{n=0}^{\infty}$, we have $M_1(\mathcal{C}_n^{(4)}) \uparrow \infty$ as $n \to \infty$ and thus $\langle \infty, \overline{M}_2 \rangle$ is a feasible partition. Since $T_1(\infty, \overline{M}_2) = \lambda_1$, this leads to a contradiction with (a.ii).

Finally in case (c.ii) it is possible to construct a subsequence $\{\mathcal{C}_n^{(5)}\}_{n=0}^{\infty}$ of $\{\mathcal{C}_n^{(3)}\}_{n=0}^{\infty}$ with $M_2(\mathcal{C}_n^{(5)})$ strictly increasing to ∞ for $n \in \mathbb{N}$. Since by construction $M_1(\mathcal{C}_n^{(3)})$ and thus also $M_1(\mathcal{C}_n^{(5)})$ is strictly increasing, we may conclude that $\langle \infty, \infty \rangle$ is a feasible partition. Since $T_1(\infty, \infty) = \lambda_1$, we get again a contradiction with (a.ii). As a result we may thus conclude that there exists an optimal partition. □

With this lemma we can now proceed with sufficient conditions for the existence of feasible partitions. The first sufficient condition for existence concerns a queueing model where the parameters ρ_1 and ρ_2 and the bound S are chosen such that the constraint on S_1 is satisfied by all partitioning policies.

LEMMA 5.3.4 *If the queue is ergodic for the partitioning policy with thresholds* $M_1 = M_2 = \infty$, *i.e.* $\rho_1 + \rho_2 < 1$, *and if*

$$S \geq \frac{1}{\mu_1(1-\rho_1-\rho_2)} \tag{5.26}$$

then $M_1 = M_2 = \infty$ *is the optimal partitioning policy.*

PROOF. Note that if $\rho_1+\rho_2 < 1$, then the right hand side of (5.26) is the expression for $S_1(\infty,\infty)$ (cf. REISER AND KOBAYASHI [72, equation (37)]). If the inequality (5.26) holds, then we know from Lemma 5.3.2 that $S_1(M_1,M_2) \leq S$ for all M_1, $M_2 \in \mathbb{N}$. Furthermore $T_i(M_1,M_2) \leq \lambda_i$ for M_1, $M_2 \in \mathbb{N}$ and $i = 1, 2$, and this bound is tight for $M_1 = M_2 = \infty$. Therefore $\langle\infty,\infty\rangle$ is a feasible policy, since $T_2(\infty,\infty) \geq T$. The class 1 throughput $T_1(\infty,\infty) = \lambda_1$, so $\langle\infty,\infty\rangle$ is the optimal partition. □

The following two lemmas give necessary and sufficient conditions for less trivial parameter settings.

LEMMA 5.3.5 *If the lower bound* $T = \lambda_2$, *then there exists an optimal partition if and only if* $S_1(1,\infty) \leq S$.

PROOF. Observe that from (5.5) the throughput $T_2(\mathcal{C})$ of class 2 under policy $f_{\mathcal{C}}$ is equal to the arrival rate λ_2 times the probability that a class 2 customer is admitted under $f_{\mathcal{C}}$. Since the blocking probability for class 2 is zero only for partitions with threshold $M_2 = \infty$, a feasible policy must thus use an infinite class 2 threshold. If $S_1(1,\infty) > S$, then by the inequality (5.24) we have $S_1(M_1,\infty) > S$ for all $M_1 \in \mathbb{N}$ and thus no feasible partition exists. If $S_1(1,\infty) \leq S$, then there exists at least one feasible partition, viz. $\langle 1,\infty\rangle$, and thus there exists an optimal partition. □

LEMMA 5.3.6 *If* $T < \lambda_2$, *then there exists an optimal partition if and only if* $S \geq S_1(1,K)$, *where* K *is defined as*

$$K = \min\{\, M_2 \in \mathbb{N} \mid T_2(1,M_2) \geq T \,\}. \tag{5.27}$$

PROOF. Let $T < \lambda_2$ and let K be defined as in (5.27). Such a K exists and is finite since $T_2(1,M_2) \uparrow \lambda_2$ as $M_2 \to \infty$. From the monotonicity of T_2 we know that a partition $\langle M_1,M_2\rangle$ can be feasible only if M_2 is at least K. If $S_1(1,K) \leq S$, then $\langle 1,K\rangle$ is a feasible partition and thus an optimal policy exists. If $S_1(1,K) > S$, then by (5.24) for all $M_1 \geq 1$ and $M_2 \geq K$ we have $S_1(M_1,M_2) > S$, and consequently there are no feasible policies. □

We can derive closed form expressions for $S_1(1,M_2)$ and $T_2(1,M_2)$, $M_2 \in \mathbb{N}$, by employing the relations (5.4)–(5.8). If we define for $n \in \mathbb{N}$

$$J_4(n) = \sum_{j=0}^{n} \rho_2^j = \frac{1-\rho_2^{n+1}}{1-\rho_2}, \tag{5.28}$$

and

$$J_5(n) = \sum_{j=0}^{n} (j+1)\rho_1\rho_2^j = \rho_1 \frac{1-(n+2)\rho_2^{n+1}+(n+1)\rho_2^{n+2}}{(1-\rho_2)^2}, \tag{5.29}$$

one can show by (5.5)–(5.8) that

$$S_1(1, M_2) = \frac{J_5(M_2)}{\lambda_1(J_4(M_2) + J_5(M_2))} \tag{5.30}$$

and

$$T_2(1, M_2) = \lambda_2 \frac{J_4(M_2 - 1) + J_5(M_2 - 1)}{J_4(M_2) + J_5(M_2)}. \tag{5.31}$$

The expressions (5.30) and (5.31) can be evaluated by a recursion in n for $J_4(n)$ and $J_5(n)$. The monotonicity of S_1 and T_2 can be employed to search for a K satisfying (5.27).

5.3.3 Design of the optimal partition

Once the existence of a feasible partition has been established, we can use the monotonicity properties of Lemma 5.3.2 to design a procedure to find the optimal partition. Assume for instance that we have a partition $\langle M_1, M_2 \rangle$ that violates the constraint $S_1(M_1, M_2) \leq S$. From the monotonicity of S_1 we then know that a feasible partition $\langle M_1^*, M_2^* \rangle$ must satisfy $M_i^* \leq M_i$, $i = 1, 2$.

Since we have expressions for all the performance measures that are involved in the formulation of problem (P_P), we are all set to discuss the algorithm to find the optimal partition. In this procedure we can distinguish the following steps.

(i) Test if there exists a feasible partition.

(ii) Determine the set of feasible partitions.

(iii) Restrict the set of feasible partitions to a finite set that contains the optimal partition.

(iv) Find the optimal partition over this finite set.

It is not necessary in all cases to perform all of these four steps, since for some examples we can already conclude in steps (i) or (ii) what the optimal partition is (if any).

For step (i) we use the results that were obtained in Lemma's 5.3.4–5.3.6. When we have a situation where Lemma 5.3.4 applies, then obviously we can stop, since we have found an optimal partition. If this is not the case, then we can check whether we have an example of Lemma 5.3.5. If this is true, then we know from Lemma 5.3.5 that the optimal partition must be of the form $\langle M_1, \infty \rangle$ for some $M_1 \in \mathbb{N}$. Note that the value of $S_1(M_1, \infty)$ can be computed from Lemma 5.3.1. If there exists a feasible partition, then clearly the optimal partition is $\langle M_1^*, \infty \rangle$, with

$$M_1^* = \max\{ M_1 \in \mathbb{N} \mid S_1(M_1, \infty) \leq S \}.$$

Such an M_1^* exists and is finite, since $S_1(M_1, \infty) \uparrow \infty$ as $M_1 \to \infty$ (otherwise the conditions of Lemma 5.3.4 would have been satisfied).

If we can exclude the cases where Lemma 5.3.4 and 5.3.5 apply, then we are in a situation where at least one of the two thresholds of the optimal partition must be finite. We can then characterise the set of feasible policies. We do this by determining for any fixed value of M_2 the maximally feasible class 1 threshold. We define for all $M_2 \in \mathbb{N} \cup \{\infty\}$

$$\mathcal{M}_1^T(M_2) = \sup\{ M_1 \in \mathbb{N} \mid T_2(M_1, M_2) \geq T \},$$

$$\mathcal{M}_1^S(M_2) = \sup\{ M_1 \in \mathbb{N} \mid S_1(M_1, M_2) \leq S \},$$

$$\mathcal{M}_1(M_2) = \min\{ \mathcal{M}_1^T(M_2), \mathcal{M}_1^S(M_2) \}.$$

For any fixed M_2 a partition $\langle M_1, M_2 \rangle$ can be feasible only if $M_1 \leq \mathcal{M}_1(M_2)$. An immediate result from the monotonicity properties is the local optimality of the class 1 threshold $\mathcal{M}_1(M_2)$ for fixed M_2, since for that threshold the maximal class 1 throughput is obtained.

The set of all feasible partitions can now be characterised as

$$\{ \langle M_1, M_2 \rangle \mid M_2 \in \mathbb{N} \cup \{\infty\}, 0 \leq M_1 \leq \mathcal{M}_1(M_2) \}.$$

and for optimisation purposes this set can be restricted to

$$\{ \langle \mathcal{M}_1(M_2), M_2 \rangle \mid M_2 \in \mathbb{N} \cup \{\infty\} \},$$

that contains the optimal partition. Examination of the structure of this set reveals that there are two difficulties in the evaluation of T_1 over this set. First we see that we have infinitely many values for M_2 and second we can encounter the situation where $\mathcal{M}_1(M_2) = \infty$ for some M_2. With respect to the latter possibility we can remark that this is in fact no difficulty, since if for some M_2 the partition $\langle \infty, M_2 \rangle$ is feasible, then it is also optimal since $T_1(\infty, M_2) = \lambda_1$.

The remaining problem thus concerns the infinitely many values that M_2 can take. We shall show that we can restrict ourselves to a finite set, thus reducing the search of the optimal partition to a search over a finite set. For this we need the following lemma.

LEMMA 5.3.7

(i) $\mathcal{M}_1^T(m)$ *is non-decreasing in* m.

(ii) $\mathcal{M}_1^S(m)$ *is non-increasing in* m.

(iii) $\mathcal{M}_1(m)$ *is unimodal in* m, *i.e.* $\mathcal{M}_1(m-1) \geq \mathcal{M}_1(m)$ *implies* $\mathcal{M}_1(m) \geq \mathcal{M}_1(m+1)$, *and* $\mathcal{M}_1(m) \leq \mathcal{M}_1(m+1)$ *implies* $\mathcal{M}_1(m-1) \leq \mathcal{M}_1(m)$.

PROOF. Parts (i) and (ii) are immediate from the monotonicity of T_2 and S_1, respectively. Part (iii) follows from the definition of $\mathcal{M}_1$ and (i) and (ii). □

Recall from the previous steps of the algorithm, that we can exclude the possibility of a feasible partition with two infinite thresholds. We may therefore conclude by the unimodality of $\mathcal{M}_1$ that $\mathcal{M}_1(m)$ converges to a finite value $\mathcal{M}_1(\infty)$ as $m \to \infty$. This limit can be computed from the expressions for $S_1(M_1, \infty)$ and $T_2(M_1, \infty)$. Since $\mathcal{M}(m)$ can thus take only finitely many values in $\mathbb{N} \cup \{\infty\}$, there must be a finite $\mathcal{M}_2$ such that $\mathcal{M}_1(M_2) = \mathcal{M}_1(\infty)$ for $M_2 \geq \mathcal{M}_2$. From the monotonicity of T_1 it is clear that within the set

$$\{ \langle \mathcal{M}_1(M_2), M_2 \rangle \mid M_2 \geq \mathcal{M}_2 \}$$

the maximum class 1 throughput is obtained for partition $\langle \mathcal{M}_1(\mathcal{M}_2), \mathcal{M}_2 \rangle$. Therefore it is sufficient to restrict the search for the optimal partition to the finite set

$$\{\langle \mathcal{M}_1(M_2), M_2\rangle \mid 0 \le M_2 \le \mathcal{M}_2\}.$$

Summing up the algorithm we thus have:

ALGORITHM 5.3.8

(i) *If* $\rho_1+\rho_2 < 1$ *and* $S \ge \big(\mu_1(1-\rho_1-\rho_2)\big)^{-1}$, *then the optimal partition is* $\langle\infty,\infty\rangle$. *Otherwise continue with step (ii).*

(ii) *If* $T = \lambda_2$ *and* $S_1(1,\infty) > S$, *then no feasible policy exists. If* $T = \lambda_2$ *and* $S_1(1,\infty) \le S$, *then the optimal policy is* $\langle M_1^*,\infty\rangle$, *where* M_1^* *can be computed with the aid of Lemma 5.3.1 as*

$$M_1^* = \max\{ M_1 \in \mathbf{N} \mid S_1(M_1,\infty) \le S\}.$$

If the above conditions do not apply, then proceed with step (iii).

(iii) *Compute with (5.31) the value of* K *that satisfies*

$$K = \min\{ M_2 \in \mathbf{N} \mid T_2(1,M_1) \ge T\}.$$

Compute $S_1(1,K)$ *from (5.30). If* $S_1(1,K) > S$, *then no feasible partition exists. Otherwise continue with Algorithm 5.3.9.*

ALGORITHM 5.3.9
Define $\mathcal{M}_1 : \mathbf{N}\cup\{\infty\} \to \mathbf{N}\cup\{\infty\}$ *as*

$$\mathcal{M}_1(M_2) = \sup\{ M_1 \mid S_1(M_1,M_2) \le S\,,\ T_2(M_1,M_2) \ge T\}.$$

(i) *Compute* $\mathcal{M}_1(\infty)$ *with the aid of Lemma 5.3.1, and compute* $\mathcal{M}_2$, *that is defined as*

$$\mathcal{M}_2 = \min\{ M_2 \mid \mathcal{M}_1(M_2) = \mathcal{M}_1(\infty)\}.$$

(ii) *Let* $M_2 = 0$.

(iii) *Compute* $\mathcal{M}_1(M_2)$ *and* $T_1(\mathcal{M}_1(M_2),M_2)$. *If for some* M_2 *we get* $\mathcal{M}_1(M_2) = \infty$, *then* $\langle\infty,M_2\rangle$ *is an optimal partition. If* $M_2 = \mathcal{M}_2$, *then proceed with step (iv). Otherwise increase* M_2 *by 1 and repeat step (iii).*

(iv) *Determine the value of* M_2, $0 \le M_2 \le \mathcal{M}_2$ *that attains the maximum value for* $T_1(\mathcal{M}_1(M_2),M_2)$.

5.4 Sharing policies

In this section we shall briefly discuss sharing policies. We know from Theorem 3.2.10 that for sharing policies the equilibrium distribution is given by the product form (5.4). This means that the performance measures can be evaluated quite easily. The main part of the algorithm is the computation of the normalisation constant $G(\mathcal{C})$. We shall not discuss the evaluation algorithm here, but refer to KAUFMAN [49, Section V].

In contrast with the partitioning policies, there does not exist monotonicity of the performance measures with respect to the weight factors c_1, c_2, and the poolsize c. This can be proven explicitly for some simple sharing policies. Consider for example a sharing policy with weight factors $c_1 = 2$ and $c_2 = 3$. If the poolsize $c = 2$, then the admission region consists of the states $(0,0)$ and $(1,0)$. This means that according to (5.1) a class 1 customer will be admitted to the queue only if it is empty, so the class 1 throughput in this case is

$$\lambda_1 \frac{\bar{\pi}(0,0)}{\bar{\pi}(0,0) + \bar{\pi}(1,0)} = \frac{\lambda_1}{1+\rho_1}.$$

Increasing the poolsize to $c = 3$ adds the state $(0,1)$ to the admission region. The effect is that class 1 customers are still admitted only when they arrive to an empty queue. The class 1 throughput for $c = 3$ is thus

$$\lambda_1 \frac{\bar{\pi}(0,0)}{\bar{\pi}(0,0) + \bar{\pi}(1,0) + \bar{\pi}(0,1)} = \frac{\lambda_1}{1+\rho_1+\rho_2},$$

which is smaller than the throughput for $c = 2$. If we increase the poolsize to $c = 4$, then the state (2,0) is added to the admission region. Now class 1 customers will also be admitted to the queue if the state is (1,0) at the moment of arrival. The class 1 throughput for $c = 4$ is thus

$$\lambda_1 \frac{\bar{\pi}(0,0) + \bar{\pi}(1,0)}{\bar{\pi}(0,0) + \bar{\pi}(1,0) + \bar{\pi}(0,1) + \bar{\pi}(2,0)} = \lambda_1 \frac{1+\rho_1}{1+\rho_1+\rho_2+\rho_1^2},$$

which is larger than T_1 for $c = 3$.

The lack of monotonicity for less trivial examples can be illustrated by Figures 5.5–5.7 where the throughputs T_1, T_2, and the expected sojourn time S_1 of a sharing policy are depicted as a function of the poolsize. The system under consideration is a queue with parameters $\lambda_1 = \lambda_2 = 0.5$, $\mu_1 = \mu_2 = 1.0$, and weight factors $c_1 = 5$, $c_2 = 9$. In fact the only performance measure that exhibits monotonicity is the expected class 1 sojourn time S_1. This should not come as a surprise, since an increase of the poolsize will always lead to more customers being admitted and a decrease of the processor's idle time. We have not been able, however, to prove this monotonicity property.

With regard to the throughputs T_1 and T_2, one can observe in Figures 5.5 and 5.6 that T_i, $i = 1, 2$, is monotone increasing if the poolsize is increased by a multiple of the weight factor c_1. This observation and the fact that T_i does not necessarily increase when the poolsize is increased by one, can partly be explained from Figure 5.8. Compare for example the admission regions for $c = 17$ and $c = 18$. The regions differ only in one state, viz. $\mathbf{m} = (0,2)$, and this extra state is beneficial only for class 2 customers, since it does not change the admission policy for class 1. Consequently class 2 throughput is larger for $c = 18$ than for $c = 17$, while the opposite is true for T_1. A reverse effect can be observed in increasing c from 19 to 20. Note also that for $c = 20, 21, 22$, the admission regions are identical and so are the performance measures.

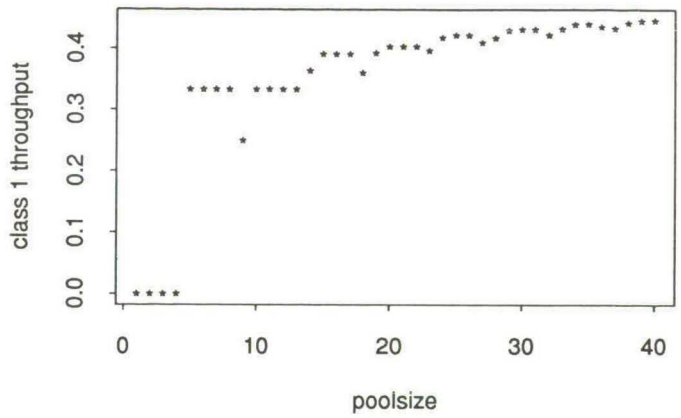

FIGURE 5.5. Throughput of class 1 as a function of the poolsize. Parameters $\lambda_1 = \lambda_2 = 0.5$, $\mu_1 = \mu_2 = 1.0$, weight factors $c_1 = 5$, $c_2 = 9$.

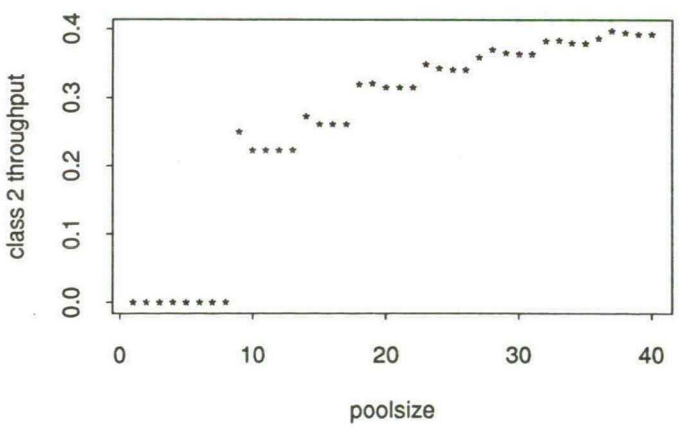

FIGURE 5.6. Throughput of class 2 as a function of the poolsize. Parameters $\lambda_1 = \lambda_2 = 0.5$, $\mu_1 = \mu_2 = 1.0$, weight factors $c_1 = 5$, $c_2 = 9$.

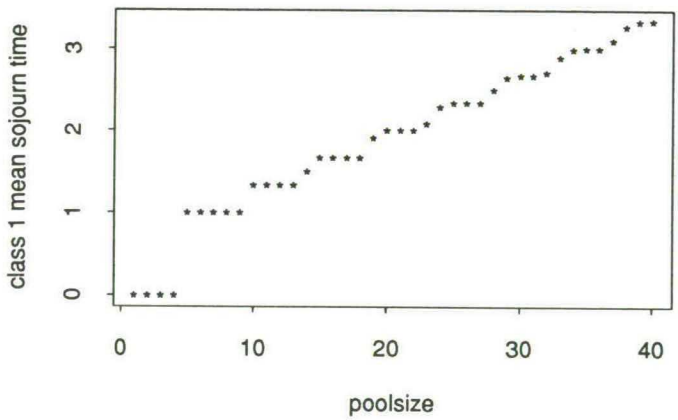

FIGURE 5.7. Mean sojourn time of class 1 as a function of the poolsize. Parameters $\lambda_1 = \lambda_2 = 0.5, \mu_1 = \mu_2 = 1.0$, weight factors $c_1 = 5, c_2 = 9$.

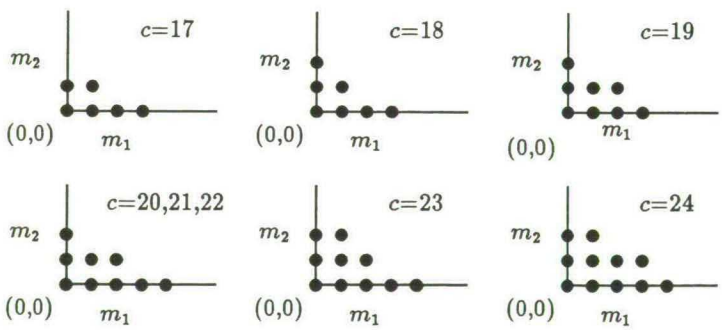

FIGURE 5.8. Admission region as a function of the poolsize c. Weight factors $c_1 = 5$, $c_2 = 9$.

5.5 Numerical results

In this section we shall discuss the performance of partitioning and sharing policies from numerical examples. For several parameter settings of a queueing model the optimal partitions are computed and the robustness of this class of policies is discussed. For the same set of parameters we compute sharing policies of which the performance is close to the corresponding optimal partitioning policies. The robustness of sharing policies is also included in the exposition.

The queueing system we consider has class 2 arrival rate $\lambda_2 = 0.3$, and service rates $\mu_1 = \mu_2 = 1.0$. The lower bound on the class 2 throughput is set at $T = 0.2$ and the upper bound on the mean class 1 sojourn time is $S = 20$. In Table 5.1 the values of the optimal partitions' thresholds are presented for different values of the class 1 arrival rate λ_1. Included in the table are the values of the performance measures for these partitions. The third column in the table refers to the so-called *normalised arrival rate* $\bar{\lambda}_1$. Due to the lower bound $T = 0.2$ on the class 2 throughput, only a fraction 0.8 of the processor's speed is available to serve class 1 customers. Since $\mu_1 = 1.0$, this means that $\lambda_1 = 0.8$ corresponds to a call request load of 100%. For this reason the normalised arrival rate is introduced, and it is defined as $\bar{\lambda}_1 = \lambda_1/0.8$.

From Lemma 5.3.4 we know that for $\lambda_1 \leq 0.65$ ($\bar{\lambda}_1 \leq 0.81$) the mean sojourn time of class 1 will always be smaller than 20, and thus partition $\langle \infty, \infty \rangle$ is feasible and optimal. For $\lambda_1 \leq 0.80$ ($\bar{\lambda}_1 \leq 1.00$) the optimal class 1 threshold is infinite. We have not included these policies in the results, since they are uninteresting when we come to the robustness issue. Because of the infinite threshold, the queue will no longer be ergodic when $\bar{\lambda}_1 \geq 1.00$.

In Figures 5.9–5.11 the performance measures for the six partitions of Table 5.1 are depicted when the class 1 arrival rate is varied while the thresholds are kept constant. The numbers in the figures correspond to the numbers of the partitions in Table 5.1. From Figure 5.9 we can conclude that the largest values of T_1 can be obtained by using partition No. 1 — optimal for $\bar{\lambda}_1 = 1.0$ — for all arrival rates. This observation can be explained immediately from the monotonicity properties of Lemma 5.3.2. Due to the large thresholds of this partition the bounds on T_2 and S_1 are violated for large $\bar{\lambda}_1$, however. Conversely, partition No. 6 that is optimal for $\bar{\lambda}_1 = 1.5$ satisfies the constraints for all values of $\bar{\lambda}_1 \in [0.0, 4.0]$ but gives for $\bar{\lambda}_1 = 1.0$ a class 1 throughput that is about 7% less than the optimal value at that load. If one considers this loss to be too large, then clearly partitioning policies are not robust.

Next we discuss the performance of sharing policies. We have considered the processor sharing queueing model for the same parameter settings as the partitioning policies. From Section 5.4 we know that sharing policies do not exhibit monotonicity, that can be used to search for optimal policies. The performance measures, however, can be computed quite efficiently, thus allowing an "interactive" search for satisfying policies, i.e. policies of which the performance is close to the optimal partitioning policies. In the examples we found that the following rules can be applied in the search for a satisfying sharing policy if $\bar{\lambda}_1 \geq 1.0$. First we have to search for the right ratio of the weight factors to get the throughputs T_1 and T_2 close to 0.8 and 0.2, respectively. Having found the right ratio, we then have to vary the poolsize c until S_1 is close to 20. In most cases we were able to find sharing policies of which the performance was close to that of partitioning policies (see Table 5.2). From this table we can conclude that when $\bar{\lambda}_1$ increases, then the ratio of c_1 and c_2 has to shift to give class 2 a preference

No.	λ_1	$\bar{\lambda}_1$	M_1^{opt}	M_2^{opt}	T_1	T_2	S_1
1	0.80	1.00	32	5	0.788	0.201	19.7
2	0.85	1.06	23	5	0.792	0.202	18.8
3	0.90	1.13	20	5	0.794	0.203	18.2
4	1.00	1.25	18	5	0.798	0.201	18.2
5	1.20	1.50	16	5	0.797	0.203	17.7
6	1.50	1.88	15	5	0.796	0.204	17.5

TABLE 5.1. OPTIMAL PARTITIONING POLICIES.

over class 1 (compare policies No. 1 and No. 6).

Analogously to the investigation on partitioning policies we have examined the robustness of sharing policies. The results of these computations are shown in Figures 5.12–5.14. From these figures we can draw the same conclusions as for the partitioning policies, viz. the policy that is optimal for $\bar{\lambda}_1 = 1.0$ violates the constraints when the arrival intensity increases. In this case also the sharing policy that is optimal for $\bar{\lambda}_1 = 1.5$ is not feasible for $\bar{\lambda}_1 = 4.0$. The loss of class 1 throughput at $\bar{\lambda}_1 = 1.00$ of sharing policy No. 6 compared to No. 1 is 9%, so sharing policies are not robust either. For some sharing policies the class 2 throughput even decreases to zero when the load becomes high. This observation can be explained from Figure 5.8. If the weight factors are such that $c_1 > c_2$, then the admission region contains exactly one state with a maximal class 1 queue length. If the queue becomes overloaded, then the equilibrium distribution will have almost all of its mass in this state. Since this state has a zero class 2 queue length, hardly any class 2 customers will be admitted and consequently T_2 becomes zero.

For both classes of admission policies we conjecture that the performance under varying loads can be improved by an adaptive control scheme. There are two approaches to construct adaptive partitioning or sharing policies. In the first approach we determine the optimal or satisfying policies for a number of values of λ_1. In the exchange the value of this parameter is estimated periodically and the policy is adjusted according to the latest estimate. This type of control is called *certainty equivalence adaptive control.*

In the second approach we use one or several system characteristics as indicators for the value of λ_1. As an example one can think of the queue length, since a large number of customers may indicate that the arrival rate is high. In this approach the admission policy should be adjusted periodically according to the actual queue length.

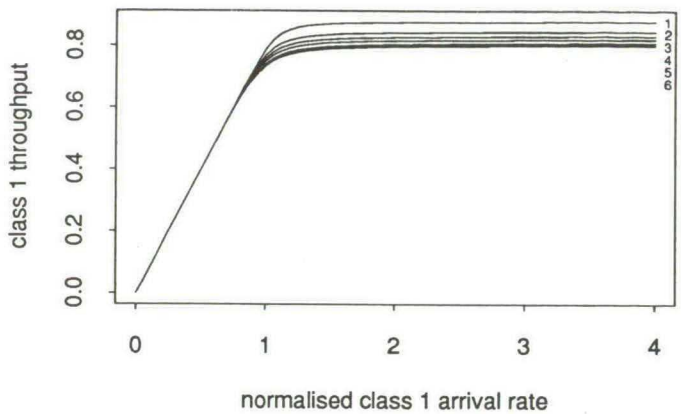

FIGURE 5.9. Throughput of class 1 for partitioning policies.

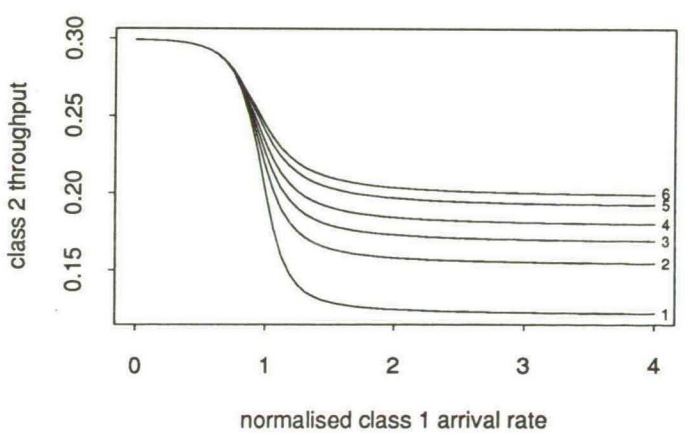

FIGURE 5.10. Throughput of class 2 for partitioning policies.

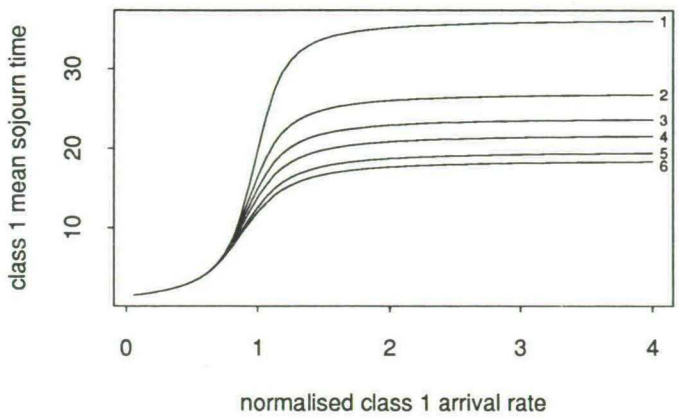

FIGURE 5.11. Mean sojourn time of class 1 for partitioning policies.

No.	λ_1	$\bar{\lambda}_1$	c_1	c_2	c	T_1	T_2	S_1
1	0.80	1.00	1	15	80	0.781	0.200	15.2
2	0.85	1.06	1	6	50	0.795	0.200	19.8
3	0.90	1.13	1	3	34	0.793	0.205	19.7
4	1.00	1.25	4	7	105	0.798	0.201	19.3
5	1.20	1.50	12	11	263	0.799	0.201	20.0
6	1.50	1.88	5	3	99	0.797	0.203	19.9

TABLE 5.2. SATISFYING SHARING POLICIES.

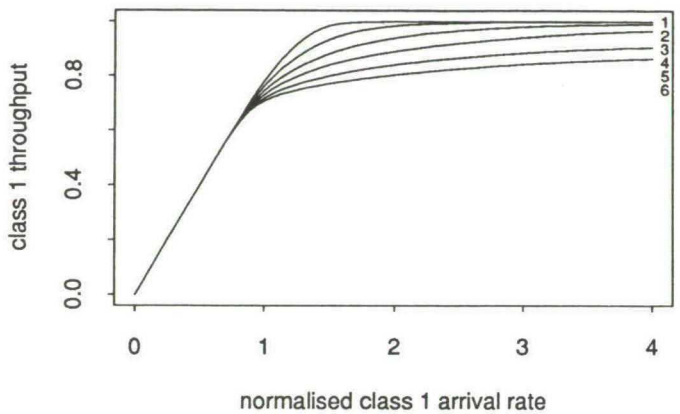

FIGURE 5.12. Throughput of class 1 for sharing policies.

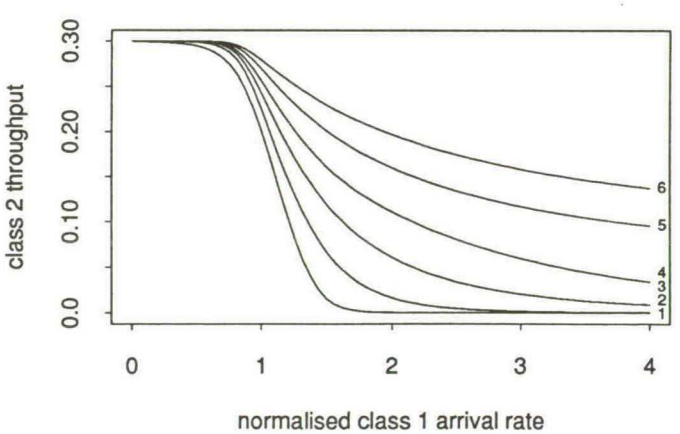

FIGURE 5.13. Throughput of class 2 for sharing policies.

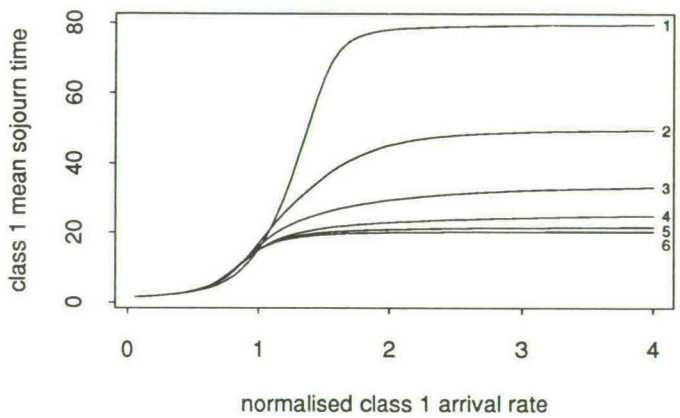

FIGURE 5.14. Mean sojourn time of class 1 for sharing policies.

6

Conclusions

In this thesis we have presented both synthesis and analysis of overload control algorithms for telephone exchanges. The main objective of the algorithms was to ensure a good level of service, even under overload conditions. We have tried to capture the ill-behaved performance of uncontrolled exchanges in queueing models. For these models we have borrowed methodologies from performance analysis and optimal stochastic control to study the effects of several control algorithms.

In the M/M/c queue that we described in Chapter 4, we succeeded in constructing an elementary queueing model with impatient customers, of which the performance agreed with the typical SPC's behaviour. Furthermore we were able to synthesise and design a simple control algorithm that improved this performance considerably and proved to be robust against fluctuations in the offered load. As a bonus we developed a new technique for the synthesis problem of optimal control in queues.

In the Processor Sharing queue of Chapter 5 we captured the effect of call processing and operator tasks on an SPC's performance. We analysed the performance of two classes of control algorithms and for one class we were able to solve a constrained optimisation problem. Unfortunately the robustness against load fluctuations of both classes of algorithms left much to be desired. This problem should be investigated in future research, where attention should be paid to the development of adaptive control algorithms.

The kind of research that has been presented in this thesis usually leads to many new problems, the above mentioned problem being just one example. We shall conclude this chapter with some suggestions for future research.

The new technique that was introduced in Chapter 4 seems to offer an exciting new method for those control synthesis problems in queues, that are hard to solve with the methodologies mentioned in Section 3.3.4. At the moment when this thesis is going to press, we have succeeded in solving a control problem that is closely related to the problem in Chapter 4. The problem deals with the same M/M/c queue that is now being offered an additional arrival stream of customers who are always allowed

to enter the queue. This second stream is introduced to model call requests that are forwarded from another exchange. Since these customers have already received service (at the originating exchange), it seems justifiable always to accept these requests. Taking the same impatience or deadline structure for the first arrival stream, we have been able to show that the optimal admission control is again of threshold type. These investigations shall be published in the near future in DE WAAL *et al.* [103]. Other problems for which we expect the new method to be fruitful are scheduling problems in queues and the M/M/1 Processor Sharing and GI/M/1 FCFS queue with impatient customers.

Apart from these problems, that stem from practical control applications, there remains a number of unsolved synthesis problems of a more theoretical nature. The results on structural properties of optimal control laws that are available in literature usually deal with queueing models that have a state space of small dimension. Structural properties of optimal control laws are usually formulated as monotonicity properties, like convexity or unimodality, of the optimal value function. There are two reasons why the extension of these results to higher dimensions appears to be difficult. First it is not sufficient to transform a one dimensional monotonicity property into an equivalent property for a higher dimension. For example the few results on synthesis of optimal control laws in queueing systems with state space dimension equal to two require a more restrictive form of monotonicity than convexity, which is usually sufficient in one dimensional problems. A first attempt made in this direction is the notion of submodularity used in BARTROLI AND STIDHAM [5]. A second reason why the extension to complex networks is difficult is inherent to the methods that are used for the derivation of structural properties. In the backward recursion method of Section 3.3.4.1 we encounter the problem that was mentioned before, viz. the stronger forms of monotonicity needed for the optimal value function. It is not always clear beforehand which properties are needed to be able to make the induction step in this method. Interchange arguments, as described in Section 3.3.4.2, are difficult to apply in complex models, since an interchange of two events may result in two sample paths that differ too much to allow an easy comparison.

Another challenging problem concerns overload control in a communication network. We predict that in the coming years this problem will become important due to the growth of networks and the increasing complexity of communication services. The upcoming introduction of *broadband ISDN* (Integrated Services Digital Network) and *ATM* (Asynchronous Transfer Mode) networks not only offers new means to implement sophisticated control algorithms, but will cause overload behaviour that may be fundamentally different from that in telephone exchanges. The many different services that shall be offered on this network will give rise to new traffic and service requirement characteristics.

Appendix A

Product form results

In this appendix we present the proof of Theorem 3.2.10 plus some extensions to the queueing model.

A.1 Proof of the main theorem

Recall Theorem 3.2.10:

THEOREM A.1.1 *Assume that we have a queueing network of two clusters of stations, where the departure and arrival blocking functions satisfy Conditions 3.2.4 and 3.2.6. Assume that the service time distributions at FCFS stations are independent of the customer class and exponentially distributed. Let M_0 be the C_2-population vector at time 0, then the equilibrium probability $\pi(\mathbf{X})$ of being in state $\mathbf{X}$, that is characterised by a C_2-population vector $M_2 \in \mathcal{E}$, is*

$$\pi(\mathbf{X}) = G\,F(M_2) \prod_{n=1}^{N} \prod_{s=1}^{k_n} \frac{\theta_n^{x_{ns}}}{f_n(s)} \left(1 - B_n^{x_{ns}}(r_{ns})\right) \tag{A.1}$$

where G is a normalisation constant.

PROOF. We prove that (A.1) satisfies the forward Kolmogorov differential equations. These state that the global probability flow out of a state should equal the flow into the state. They can be written as

$$0 = \sum_{n} \sum_{s} \Xi(\mathbf{X}, n, s) \tag{A.2}$$

where $\Xi(\mathbf{X}, n, s)$ is a local probability flow:

$$\Xi(\mathbf{X}, n, s) := \frac{\partial \pi(\mathbf{X})}{\partial r_{ns}} f_n(k_n) \phi_n(k_n, s) + \tag{A.3a}$$

$$+\sum_{\substack{m\in C_1\\ m\neq n}}\sum_{t=1}^{k_m^X+1}\pi\big(\mathbf{X}-(n,s)+(m,t,0^+)\big)\phi_m(k_m^X+1,t)f_m(k_m^X+1)\times$$

$$\times\delta_n(k_n-1,s)\beta_n^{x_{ns}}(r_{ns})\Big\{R_{mn}^{x_{ns}}+R_{m0}^{x_{ns}}\big(1-A^{x_{ns}}(M_2)\big)R_{0n}^{x_{ns}}\Big\} \qquad \text{(A.3b)}$$

$$+\quad\sum_{t=1}^{k_n}\pi\big(\mathbf{X}-(n,s)+(n,t,0^+)\big)\phi_n(k_n,t)f_n(k_n)\times$$

$$\times\delta_n(k_n-1,s)\beta_n^{x_{ns}}(r_{ns})\Big\{R_{nn}^{x_{ns}}+R_{n0}^{x_{ns}}\big(1-A^{x_{ns}}(M_2)\big)R_{0n}^{x_{ns}}\Big\} \qquad \text{(A.3c)}$$

$$+\sum_{m\in C_2}\sum_{t=1}^{k_m^X+1}\pi\big(\mathbf{X}-(n,s)+(m,t,0^+)\big)\phi_m(k_m^X+1,t)f_m(k_m^X+1)\times$$

$$\times\delta_n(k_n-1,s)\beta_n^{x_{ns}}(r_{ns})\Big\{R_{m0}D^{x_{ns}}(M_2+\mathbf{e}_{x_{ns}})R_{0n}^{x_{ns}}\Big\} \qquad \text{(A.3d)}$$

for $n\in C_1$ and analogously for $n\in C_2$. In this definition of Ξ (A.3a) corresponds to a probability flow out of state $\mathbf{X}$ due to a change in the residual service time r_{ns} of the job at position (n,s). The terms (A.3b), (A.3c) and (A.3d) represent flows into state $\mathbf{X}$ due to an arrival of a customer from queue $m\in C_1$, $m\neq n$, from queue n and from queue $m\in C_2$, respectively.

PROOF FOR SYMMETRIC QUEUES. First we prove that if n is a symmetric queue, then $\pi(\cdot)$ as in (A.1) satisfies the so-called *Job Local Balance* equations. These equations state that there should be a balance of the loss in probability density of state $\mathbf{X}$ due to a departure of *one* job against the gain in probability density due to an arrival of *the same* job. From the definition of Ξ it is clear that this corresponds to the equation $\Xi(\mathbf{X},n,s)=0$.

Assume that $n\in C_1$ is a symmetric queue. Note that according to (A.1)

$$\pi\big(\mathbf{X}-(n,s)+(m,t,0^+)\big)$$

$$=\begin{cases}\dfrac{\pi(\mathbf{X})f_n(k_n)\theta_m^{x_{ns}}}{\big(1-B_n^{x_{ns}}(r_{ns})\big)f_m(k_m^X+1)\theta_n^{x_{ns}}}, & m\in C_1,\ m\neq n,\\[2ex] \dfrac{\pi(\mathbf{X})}{\big(1-B_n^{x_{ns}}(r_{ns})\big)}, & m=n,\\[2ex] \dfrac{\pi(\mathbf{X})f_n(k_n)\theta_m^{x_{ns}}F(M_2+\mathbf{e}_{x_{ns}})}{\big(1-B_n^{x_{ns}}(r_{ns})\big)f_m(k_m^X+1)\theta_n^{x_{ns}}F(M_2)}, & m\in C_2,\end{cases} \qquad \text{(A.4)}$$

and

$$\frac{\partial\pi(\mathbf{X})}{\partial r_{ns}}=-\frac{\pi(\mathbf{X})}{1-B_n^{x_{ns}}(r_{ns})}\beta_n^{x_{ns}}(r_{ns}). \qquad \text{(A.5)}$$

To show that $\Xi(\mathbf{X},n,s)=0$, we distinguish three cases:

1. $M_2+\mathbf{e}_{x_{ns}}\notin\mathcal{E}$. (Non-admissible population vector)
2. $M_2+\mathbf{e}_{x_{ns}}\in\mathcal{E}$ and $A^{x_{ns}}(M_2)=0$. (Not directly admissible population vector)
3. $M_2+\mathbf{e}_{x_{ns}}\in\mathcal{E}$ and $A^{x_{ns}}(M_2)>0$. (Directly admissible population vector)

Since we are only considering flows due to changes in job (n, s), there should be no confusion when we omit the index x_{ns} from A, B, β, θ, $\mathbf{e}$ and R. To improve the notational clarity we also omit the index $\mathbf{X}$ from k_n and k_m.

1. $(M_2 + \mathbf{e} \notin \mathcal{E})$

Since $M_2 + \mathbf{e} \notin \mathcal{E}$, we must have $A(M_2) = 0$ and according to Condition 3.2.4 also $D(M_2 + \mathbf{e}) = 0$. Consequently term (A.3d) vanishes. With (A.4) the equation $\Xi(\mathbf{X}, n, s) = 0$ becomes equivalent to

$$\begin{aligned} 0 &= \phi_n(k_n, s)\theta_n \\ &+ \sum_{\substack{m \in C_1 \\ m \neq n}} \sum_{t=1}^{k_m+1} \phi_m(k_m + 1, t)\theta_m \delta_n(k_n - 1, s)\{R_{mn} + R_{m0}R_{0n}\} \\ &+ \sum_{t=1}^{k_n} \phi_n(k_n, t)\theta_n \delta_n(k_n - 1, s)\{R_{nn} + R_{n0}R_{0n}\}. \end{aligned} \tag{A.6}$$

Since $\delta_n(k_n - 1, s) = \phi_n(k_n, s)$ (n is symmetric) and $\sum_{t=1}^{k} \phi(k, t) = 1$ a sufficient condition for (A.6) is

$$\theta_n = \sum_{m \in C_1} \theta_m \{R_{mn} + R_{m0}R_{0n}\}. \tag{A.7}$$

Using the traffic equations (3.9) and Proposition 3.2.2 this condition is verified.

2. $(M_2 + \mathbf{e} \in \mathcal{E}$ and $A(M_2) = 0)$

Due to Condition 3.2.4, $A(M_2) = 0$ is equivalent to $D(M_2 + \mathbf{e}) = 0$. Substituting this in $\Xi(\mathbf{X}, n, s) = 0$, again yields (A.6).

3. $(M_2 + \mathbf{e} \in \mathcal{E}$ and $A(M_2) > 0)$

Due to Condition 3.2.4 we have $D(M_2 + \mathbf{e}) > 0$, so there exists a path from M_2 to $M_2 + \mathbf{e}$. Since $\mathbf{v}_0 := M_2$ and $\mathbf{v}_1 := M_2 + \mathbf{e}$ is such a path, we have from (3.15)

$$F(M_2 + \mathbf{e}) = F(M_2)\frac{A(M_2)}{D(M_2 + \mathbf{e})}.$$

The equation $\Xi(\mathbf{X}, n, s) = 0$ then yields

$$\begin{aligned} &\theta_n \phi_n(k_n, s) \\ &= \sum_{\substack{m \in C_1 \\ m \neq n}} \sum_{t=1}^{k_m+1} \theta_m \phi_m(k_m + 1, t)\delta_n(k_n - 1, s)\Big\{R_{mn} + R_{m0}\big(1 - A(M_2)\big)R_{0n}\Big\} \\ &+ \sum_{t=1}^{k_n} \theta_n \phi_n(k_n, t)\delta_n(k_n - 1, s)\Big\{R_{nn} + R_{n0}\big(1 - A(M_2)\big)R_{0n}\Big\} \\ &+ \sum_{m \in C_2} \sum_{t=1}^{k_m+1} \theta_m \phi_m(k_m + 1, t)\delta_n(k_n - 1, s)\Big\{R_{m0}A(M_2)R_{0n}\Big\}. \end{aligned}$$

Using the same arguments as in Case 1 this equation is implied by

$$\theta_n = \sum_{m \in C_1} \theta_m \Big\{R_{mn} + R_{m0}\big(1 - A(M_2)\big)R_{0n}\Big\} + \sum_{m \in C_2} \theta_m R_{m0}A(M_2)R_{0n} \tag{A.8}$$

or equivalently, using $R_{m0}R_{0n} = R_{mn}$ for $n \in C_1$, $m \in C_2$,

$$\theta_n = \sum_{m \in C_1 \cup C_2} \theta_m R_{mn} + \left\{\sum_{m \in C_1} \theta_m R_{m0} - \sum_{m \in C_2} \theta_m R_{m0}\right\}\big(1 - A(M_2)\big)R_{0n}. \tag{A.9}$$

This equation is true due to (3.9) and Proposition 3.2.2.

The proof of the Job Local Balance equation for a symmetric station in C_2 proceeds in a similar manner.

PROOF FOR FCFS QUEUES. Now assume that $n \in C_1$ and n is a FCFS queue. It is obvious that we cannot have Job Local Balance in this case, since all the probability flow from departures is due to the customer leaving from the first position and flow from arrivals is due to customers arriving in the last position in the station. For FCFS queues we have, however, *Station Balance*, i.e. a balance in the gain of probability density of state $\mathbf{X}$ due to arrivals in the queue against the loss in probability density due to departures from the queue, or equivalently $\sum_{s=1}^{k_n} \Xi(\mathbf{X}, n, s) = 0$. Since $\delta_n(k_n - 1, s) = 1$ iff $s = k_n$, and $\phi_n(k_n, s) = 1$ iff $s = 1$, we have $\Xi(\mathbf{X}, n, s) = 0$ for $s = 1, \ldots, k_n - 1$. This yields

$$\begin{aligned}
\sum_{s=1}^{k_n} \Xi(\mathbf{X}, n, s) = {} & \frac{\partial \pi(\mathbf{X})}{\partial r_{n1}} \beta_n(r_{n1}) f_n(k_n) \\
& + \sum_{\substack{m \in C_1 \\ m \neq n}} \sum_{t=1}^{k_m+1} \pi\big(\mathbf{X} - (n, k_n) + (m, t, 0^+)\big) f_m(k_m + 1) \phi_m(k_m + 1, t) \times \\
& \qquad \times \beta_n(r_{nk_n}) \Big\{ R_{mn} + R_{m0}\big(1 - A(M_2)\big) R_{0n} \Big\}, \\
& + \pi\big(\mathbf{X} - (n, k_n) + (n, 1, 0^+)\big) f_n(k_n) \times \\
& \qquad \times \beta_n(r_{nk_n}) \Big\{ R_{nn} + R_{n0}\big(1 - A(M_2)\big) R_{0n} \Big\} \qquad \text{(A.10)} \\
& + \sum_{m \in C_2} \sum_{t=1}^{k_m+1} \pi\big(\mathbf{X} - (n, k_n) + (m, t, 0^+)\big) f_m(k_m + 1) \phi_m(k_m + 1, t) \times \\
& \qquad \times \beta_n(r_{nk_n}) \Big\{ R_{m0} D(M_2 + \mathbf{e}) R_{0n} \Big\}
\end{aligned}$$

where the functions A, D and the routing probabilities R should be read with index x_{nk_n}. Since $B_n(\cdot)$ and $\beta_n(\cdot)$ are independent of the customer class and $B_n(r) = 1 - e^{-\mu_n r}$, we have $\beta_n(r)/(1 - B_n(r)) = \mu_n$ and (A.10) reduces to (A.7) for Case 1 and 2, and to (A.8) for Case 3. □

A.2 STOP PROTOCOL

In the queueing network that was introduced in Section 3.2.2.1, customers who were prohibited to jump from one cluster to another, were rerouted back into the originating cluster. This protocol, called the *recirculate protocol*, can be viewed as an extension of the *repeat protocol*. The latter, that is widely used in telecommunication, requires that customers repeat their service in the same station when a jump is prohibited. The product form results in the literature are all established under the repeat or, in the exponential case, recirculate protocol (cf. HORDIJK AND VAN DIJK [42], KAUFMAN [49], KELLY [50], LAM [58], SERFOZO [89], WALRAND [105], YAO AND BUZACOTT [109]). In practice, however, a total service may comprise a number of exponential stages and is no longer of exponential form. Particularly in communication, though, the recirculate protocol, which requires all service stages to be repeated, seems unrealistic.

Alternatively, the *stop protocol* is more realistic, where service of a customer is interrupted when he is blocked. In the case of exponential service times the equivalence between the stop and recirculate (repeat) protocol is intuitively obvious and has been shown (e.g. ONVURAL AND PERROS [68]). For non-exponential situations, however, this is far from obvious and generally not true.

DEFINITION A.2.1 (STOP BLOCKING PROTOCOL)
Consider the network as described in Section 3.2.2.1. When the occupancy of cluster C_2 is equal to M_2, then for all classes k the amount of service received by all class k customers in C_1 and C_2 is reduced by a factor $A^k(M_2)$ and $D^k(M_2)$ respectively.

As an example, a customer in position s at station $n \in C_1$ will now receive service at a rate $f_n(k_n)\phi_n(k_n,s)A^{x_{ns}}(M_2)$, as opposed to a service rate $f_n(k_n)\phi_n(k_n,s)$ in the model with the recirculate blocking protocol.

THEOREM A.2.2 *Assume that we have a queueing network of two clusters of stations, where the departure and arrival blocking functions satisfy Conditions 3.2.4 and 3.2.6. All stations are symmetric. Let M_0 be the C_2-population vector at time 0, then also under the stop protocol the equilibrium probability $\pi(\mathbf{X})$ of being in state $\mathbf{X}$ is given by (A.1).*

PROOF. We have to prove that (A.1) satisfies the forward Kolmogorov equations. We prove this in a similar way as in Theorem A.1.1. Note that the local probability flow $\Xi(\mathbf{X},n,s)$, as defined in (A.3), under the stop protocol is given by

$$\begin{aligned}
\Xi(\mathbf{X},n,s) &= \frac{\partial\pi(\mathbf{X})}{\partial r_{ns}} f_n(k_n)\phi_n(k_n,s)A^{x_{ns}}(M_2) \\
&+ \sum_{\substack{m\in C_1 \\ m\neq n}} \sum_{t=1}^{k_m^X+1} \pi\big(\mathbf{X}-(n,s)+(m,t,0^+)\big)\phi_m(k_m^X+1,t)f_m(k_m^X+1)\times \\
&\qquad \times\delta_n(k_n-1,s)\beta_n^{x_{ns}}(r_{ns})\big\{R_{mn}^{x_{ns}}A^{x_{ns}}(M_2)\big\} \\
&+ \sum_{t=1}^{k_n} \pi\big(\mathbf{X}-(n,s)+(n,t,0^+)\big)\phi_n(k_n,t)f_n(k_n)\times \\
&\qquad \times\delta_n(k_n-1,s)\beta_n^{x_{ns}}(r_{ns})\big\{R_{nn}^{x_{ns}}A^{x_{ns}}(M_2)\big\} \\
&+ \sum_{m\in C_2} \sum_{t=1}^{k_m^X+1} \pi\big(\mathbf{X}-(n,s)+(m,t,0^+)\big)\phi_m(k_m^X+1,t)f_m(k_m^X+1)\times \\
&\qquad \times\delta_n(k_n-1,s)\beta_n^{x_{ns}}(r_{ns})\big\{R_{m0}D^{x_{ns}}(M_2+\mathbf{e}_{x_{ns}})R_{0n}^{x_{ns}}\big\}.
\end{aligned}$$

If $A^{x_{ns}}(M_2)=0$ (Case 1 and 2 in the proof of Theorem A.1.1), then we have both $D^{x_{ns}}(M_2+\mathbf{e}_{x_{ns}})=0$ and $\Xi(\mathbf{X},n,s)=0$. If $A^{x_{ns}}(M_2)>0$, then by Lam's Condition 3.2.4 $D^{x_{ns}}(M_2+\mathbf{e}_{x_{ns}})>0$ and $\Xi(\mathbf{X},n,s)=0$ is equivalent to

$$\begin{aligned}
&\theta_n\phi_n(k_n,s)\\
&= \sum_{\substack{m\in C_1\\ m\neq n}}\sum_{t=1}^{k_m+1}\theta_m\phi_m(k_m+1,t)\delta_n(k_n-1,s)R_{mn}A(M_2)\\
&+ \sum_{t=1}^{k_n}\theta_n\phi_n(k_n,t)\delta_n(k_n,s)R_{nn}A(M_2)\\
&+ \sum_{m\in C_2}\sum_{t=1}^{k_m+1}\theta_m\phi_m(k_m+1,t)\delta_n(k_n-1,s)R_{m0}A(M_2)R_{0n}.
\end{aligned}$$

This equation is again implied by the traffic equations (3.9). □

A.3 Type changes

Like the queueing networks in Baskett *et al.* [6] and Lam [58] we can extend our networks by allowing customers to change class when jumping. Suppose that a customer of class k who completes service at station n jumps to station m and becomes a customer of class l with probability $R_{nk;ml}$. Assume that the routing matrix $[R_{nk;ml}]$ defines a Markov chain with states (n,k) that can be decomposed into K ergodic routing subchains $\mathcal{R}_1,\ldots,\mathcal{R}_K$. We denote *type* as a membership of one of the subchains and we assume the number of customers of each type to be fixed. Define population vectors $\mathcal{M}, M_1$ and M_2, where the k-th component now denotes the number of customers of type k, and define the arrival and departure functions as $A^k(\cdot)$ and $D^k(\cdot)$ for each *type* k. Service time distributions in symmetric stations are still allowed to be dependent of customer class. For this network Theorem A.1.1 still holds, but with θ_n^k obtained from slightly more involved traffic equations (cf. Lam [58]).

A.4 Network dependent service disciplines

As in Chandy and Martin [21] the service discipline functions δ and ϕ can be made totally network dependent, provided the symmetry at non-exponential stations is not violated. As in van Dijk [27] and Serfozo [89] the service capacity function $f_i(\mathbf{k})$ at station i when the state of the network is $\mathbf{k}$, can be taken to be of the form

$$f_i(\mathbf{k}) = \frac{\chi(\mathbf{k}-\mathbf{e}_i)}{\psi(\mathbf{k})}.$$

Here $\chi(\cdot)$ and $\psi(\cdot)$ are arbitrary strictly positive functions and the vector $\mathbf{k} = (k_1,\ldots,k_{N_1+N_2})$ is the station occupancy vector with k_n the number of customers at station n. These state dependent service rates lead to the stationary probability of being in state $\mathbf{X}$

$$\pi(\mathbf{X}) = G\,F(M_2)\psi(\mathbf{k})\prod_{n=1}^{N}\prod_{s=1}^{k_n}\theta_n^{x_{ns}}\left(1-B_n^{x_{ns}}(r_{ns})\right)$$

where $\mathbf{k}$ is the station occupancy vector when the microstate is $\mathbf{X}$.

Appendix B

Deadline distributions

B.1 Examples of deadline distributions

In this appendix we give some examples of deadline distributions, that make the reward function g modified unimodal. Recall the definition of g.

Definition B.1.1 *Let $g(n)$, $n \geq 1$, be the probability that a customer who is admitted to the queue, makes his deadline when the queue length, including himself, at the time of arrival is n.*

We want to show for some specific distribution functions of the deadlines, that g is modified unimodal.

Definition B.1.2 (Modified Unimodality)
A function $g : \mathbf{N} \to \mathbf{R}_+$ is called modified unimodal with parameter ψ, $0 \leq \psi < 1$, if it satisfies the following condition:

(i) $g(n)\mu_n \leq g(n+1)\mu_{n+1}$ *for* $n < c$.

(ii) $g(n+1) \leq \psi g(n)$ *for* $n \geq c$.

The first proposition deals with the case where the deadlines are deterministic.

Proposition B.1.3 *If the deadlines of customers are deterministic, then g is modified unimodal.*

Proof. Assume that the deadlines of customers are deterministic and equal to σ. Note that if $\sigma = 0$, then a customer will never meet his deadline and $g \equiv 0$, so we may restrict our attention to the case where $\sigma > 0$. Let $n < c$. Consider a customer who finds upon arrival n customers in the queue, including himself. Since this customer can be served immediately, the probability $g(n)$ that he completes his service before his extinction time is equal to the probability that his service time does not exceed

σ. This implies that $g(n) = 1 - e^{-\mu\sigma}$, $n < c$, so clearly part (i) of Definition B.1.2 is satisfied.

Let $n \geq c$. Consider the sojourn time S_{n+1} of a customer who finds $n+1$ customers in the queue upon arrival, including himself. Due to the memoryless property of the exponential distribution we have $S_{n+1} \stackrel{d}{=} Z + S_n$, where Z and S_n are independent random variables. Z represents the time spent in the queue until the next service completion and has distribution function $1 - e^{-c\mu t}$. S_n is the sojourn time of a customer with queue length n upon arrival. Let $\psi > 0$ be such that

$$1 - e^{-c\mu\sigma} \leq \psi < 1.$$

We shall show that $P(S_{n+1} \leq \sigma) \leq \psi P(S_n \leq \sigma)$. We have

$$\begin{aligned} P(S_{n+1} \leq \sigma) &= P(Z + S_n \leq \sigma) \\ &\leq P(Z \leq \sigma,\ S_n \leq \sigma) \\ &= (1 - e^{-c\mu\sigma}) P(S_n \leq \sigma) \\ &\leq \psi P(S_n \leq \sigma). \end{aligned}$$

□

The second class of functions we consider consist of distribution functions that have a failure rate that is bounded away from 0.

DEFINITION B.1.4 (FAILURE RATE)
If a non-negative random variable has a distribution function $F : \mathbf{R}_+ \to [0,1]$ *and probability density* $f : \mathbf{R}_+ \to \mathbf{R}_+$, *then the* failure rate (*also called* hazard rate) $h : \{ t \in \mathbf{R}_+ \mid F(t) < 1 \} \to \mathbf{R}_+$ *is defined as*

$$h(t) = \frac{f(t)}{1 - F(t)}.$$

If there exists a constant $\Psi > 0$ *such that*

$$\inf_{\{t \mid F(t) < 1\}} h(t) > \Psi,$$

then we say that the random variable has a failure rate that is bounded away from 0 *by* Ψ (*cf.* STOYAN [93] *and* BARLOW AND PROSCHAN [4]).

PROPOSITION B.1.5 *If the deadlines of customers have a failure rate that is bounded away from 0 by* Ψ, *then the reward function* g *is modified unimodal with parameter* $\psi = c\mu/(c\mu + \Psi)$.

PROOF. Note that for $n < c$, $g(n)$ is constant and non-negative, and $\mu_n = n\mu$, so clearly $g(n)\mu_n \leq g(n+1)\mu_{n+1}$.

Let $n \geq c$. Consider a customer who finds $n+1$ customers in the queueing system upon arrival, including himself. Denote the deadline of this customer as D and its distribution function as F_D. If we denote the sojourn time of this tagged customer as S_{n+1}, then $S_{n+1} \stackrel{d}{=} Z + S_n$. where Z and S_n are independent random variables. Z denotes the time until the next service completion and has distribution function $F_Z(t) = 1 - e^{-c\mu t}$. S_n denotes the sojourn time of a customer with n customers in the queue and we denote its distribution function as F_{S_n}. For part (ii) of Definition B.1.2 we have to prove

$$g(n) = P(S_n \leq D) \geq \frac{\Psi + c\mu}{c\mu} P(S_n + Z \leq D) = \frac{\Psi + c\mu}{c\mu} g(n+1).$$

Note that $P(S_n \leq D) = P(S_n \leq D < S_n + Z) + P(S_n + Z \leq D)$, so it is sufficient to prove

$$P(S_n \leq D < S_n + Z) \geq \frac{\Psi}{c\mu} P(S_n + Z \leq D).$$

Note that

$$P(S_n \leq D < S_n + Z) = \int_0^\infty P(s \leq D < s + Z) dF_{S_n}(s).$$

We shall prove that $P(s \leq D < s + Z) \geq \frac{\Psi}{c\mu} P(s + Z \leq D)$ for all $s \in \mathsf{R}_+$.

$$\begin{aligned}
P(s \leq D < s + Z) &= \int_s^\infty e^{-c\mu(t-s)} dF_D(t) \\
&= e^{c\mu s} \int_s^\infty e^{-c\mu t} dF_D(t) \\
&\geq \Psi e^{c\mu s} \int_s^\infty e^{-c\mu t} (1 - F_D(t)) dt \\
&= \frac{\Psi}{c\mu} \left\{ - \left[e^{-c\mu(t-s)} \left(1 - F_D(t)\right) \right]_{t=s}^\infty - \int_s^\infty e^{-c\mu(t-s)} dF_D(t) \right\} \\
&= \frac{\Psi}{c\mu} \int_s^\infty \left(1 - e^{-c\mu(t-s)}\right) dF_D(t) \\
&= \frac{\Psi}{c\mu} P(s + Z \leq D)
\end{aligned}$$

□

PROPOSITION B.1.6 *If the deadlines of customers have an Erlang distribution, then the reward function g is modified unimodal.*

PROOF. Let $m \in \mathsf{N}$, $m \geq 1$ and $\sigma > 0$. Denote $g(m, n)$ as the probability that a customer who has a deadline that is distributed according to an Erlang distribution with m phases and transition rate σ per phase, meets his deadline, when there are n customers, including himself, in the queue. Denote

$$p = \frac{c\mu}{\sigma + c\mu}, \qquad q = 1 - p = \frac{\sigma}{\sigma + c\mu},$$

$$p' = \frac{\mu}{\sigma + \mu}, \qquad q' = 1 - p' = \frac{\sigma}{\sigma + \mu}.$$

We shall show that $g(m, n)$, $m, n \in \mathsf{N}$ satisfies

$$\frac{g(m, n+1)}{g(m, n)} \leq \psi_m < 1, \qquad n \geq c, \tag{B.1}$$

where ψ_m is defined as

$$\psi_m = 1 - q^m, \qquad m \geq 1. \tag{B.2}$$

We shall prove (B.1) by induction in m. From the definition of $g(m,n)$ it is clear that $g(m,0) = 1$, $m \geq 0$ (with $n = 0$ to be interpreted as the customer having completed his service) and $g(0,n) = 0$, $n \geq 1$ ($m = 0$ meaning that the deadline has expired). It is an elementary exercise to show that the following recursion holds:

$$g(m,n) = \begin{cases} q'g(m-1,n) + p'g(m,n-1), & \text{if } 1 \leq n \leq c, m \geq 1, \\ \\ qg(m-1,n) + pg(m,n-1), & \text{if } n > c, m \geq 1. \end{cases} \tag{B.3}$$

Initial step. From $g(0,n) = 0$, $n \geq 1$ and the recursion (B.3) we have for $n \geq c$ $g(1,n+1) = pg(1,n)$, so clearly $\psi_1 = p = 1 - q < 1$ satisfies (B.1).

Induction step. Let $m \geq 1$ and assume that ψ_m as defined in (B.2) is strictly smaller than 1 and satisfies (B.1). Using the recursion we have for $n \geq c$

$$\frac{g(m+1,n+1)}{g(m+1,n)} = \frac{pg(m+1,n) + qg(m,n+1)}{g(m+1,n)}$$

$$\leq p + q\psi_m \frac{g(m,n)}{g(m+1,n)}$$

$$\leq p + q\psi_m$$

$$= \psi_{m+1}$$

where the first inequality follows from the induction assumption. The second inequality — $g(m,n) \leq g(m+1,n)$ — is immediate from the definition of $g(m,n)$, since a customer whose deadline consists of $m+1$ exponential stages will have a higher probability of completing service successfully than a customer with only m stages for its deadline. Recall that the transition rate per stage is the same for $m+1$ and m stages. □

Bibliography

[1] F. BACCELLI (1983). Modèles Probabilistes de Systèmes Informatiques Distribués. Thèse d'état, INRIA, Paris.

[2] J.S. BARAS, A.J. DORSEY, AND A.M. MAKOWSKI (1985). Two Competing Queues with Linear Costs and Geometric Service Requirements: The μc-Rule is Often Optimal. *Adv. Appl. Prob.* 17, 186–209.

[3] J.S. BARAS, D.-J. MA, AND A.M. MAKOWSKI (1985). K Competing Queues with Geometric Service Requirements and Linear Costs: The μc-Rule is Always Optimal. *Systems Control Lett.* 6, 173–180.

[4] R.E. BARLOW AND F. PROSCHAN (1981). *Statistical Theory of Reliability and Life Testing: Probability Models.* TO BEGIN WITH, Silver Spring, MD.

[5] M. BARTROLI AND S. STIDHAM, JR. (1987). Towards a Unified Theory of Structure of Optimal Policies for Control of Networks of Queues. Preprint, Dept. of Oper. Res., Univ. of North Carolina, Chapel Hill.

[6] F. BASKETT, K.M. CHANDY, R.R. MUNTZ, AND F.G. PALACIOS (1975). Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *J. Assoc. Comput. Mach.* 22, 248–260.

[7] R. BELLMAN (1957). *Dynamic Programming.* Princeton University Press, Princeton.

[8] J.L. VAN DEN BERG (1990). *Sojourn Times in Feedback and Processor Sharing Queues.* PhD thesis, Centre for Mathematics and Computer Science, Amsterdam.

[9] A.W. BERGER (1989). Overload Control Using Rate Control Throttle: Selecting Token Bank Capacity for Robustness to Arrival Rates, in *Proc. of the 28th Conf. on Decision and Control*, 2527–2529, IEEE Press, Piscataway.

[10] D.P. BERTSEKAS (1987). *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, New Jersey.

[11] D.P. BERTSEKAS AND S.E. SHREVE (1978). *Stochastic Optimal Control: The Discrete Time Case.* Academic Press, New York.

[12] P.P. BHATTACHARYA AND A. EPHREMIDES (1989). Optimal Scheduling with Strict Deadlines. *IEEE Trans. Autom. Control* AC-34, 721–728.

[13] R.K. BOEL AND J.H. VAN SCHUPPEN (1985). Overload Control for SPC Telephone Exchanges - Refined Models and Stochastic Control, in *Proceedings*

of the 3rd Bad Honnef Conference, 100–110, N. Christopeit, K. Helmes, and M. Kohlmann (eds.), Springer-Verlag, Berlin.

[14] R.K. BOEL AND J.H. VAN SCHUPPEN (1986). Overload Control for Switches of Communication Systems - A Two-phase Model for Call Request Processing, in *Proceedings of the International Seminar on Teletraffic Analysis and Computer Performance Evaluation*, 209–224, O.J. Boxma, J.W. Cohen, and H.C. Tijms (eds.), Elsevier, Amsterdam.

[15] V.S. BORKAR (1983). Controlled Markov Chains and Stochastic Networks. *SIAM J. Control and Optimization* 21, 652–666.

[16] P. BRÉMAUD (1979). Optimal Thinning of a Point Process. *SIAM J. Control and Optimization* 17, 222–230.

[17] D.Y. BURMAN, J.P. LEHOCZKY, AND Y. LIN (1984). Insensitivity of Blocking Probabilities in a Circuit-Switching Network. *J. Appl. Prob.* 21, 850–859.

[18] C. BUYUKKOC, P. VARAIYA, AND J. WALRAND (1985). The $c\mu$ Rule Revisited. *Adv. Appl. Prob.* 17, 237–238.

[19] J.P. BUZEN (1973). Computational Algorithms for Closed Queueing Networks with Exponential Servers. *Commun. ACM* 25, 527–531.

[20] K.M. CHANDY, J.H. HOWARD, AND D.F. TOWSLEY (1977). Product Form and Local Balance in Queueing Networks. *J. Assoc. Comput. Mach.* 24, 250–263.

[21] K.M. CHANDY AND J. MARTIN (1983). A Characterization of Product Form Queueing Networks. *J. Assoc. Comput. Mach.* 30, 286–299.

[22] K.L CHUNG (1967). *Markov Chains with Stationary Transition Probabilities, 2nd ed.* Springer-Verlag, New York.

[23] J.W. COHEN (1979). The Multiple Phase Service Network with Generalized Processor Sharing. *Acta Inform.* 12, 245–284.

[24] A.E. CONWAY AND N.D. GEORGANAS (1986). A New Method for Computing the Normalization Constant of Multiple-Chain Queueing Networks. *Infor.* 24, 184–198.

[25] A.E. CONWAY AND N.D. GEORGANAS (1986). RECAL - A New Efficient Algorithm for the Exact Analysis of Multiple-Chain Closed Queueing Networks. *J. Assoc. Comput. Mach.* 33, 768–791.

[26] G. DAISENBERGER, J. OEHLERICH, AND G. WEGMANN (1989). Two Concepts for Overload Regulation in SPC Switching Systems: STATOR and TAIL. *Telecommunication Journal* 56, 306–313.

[27] N.M. VAN DIJK (1989). Product Forms for Queueing Networks with Limited Clusters. Research Memorandum 1989-29, Free University, Amsterdam.

[28] R.L. DISNEY AND D. KÖNIG (1985). Queueing Networks: A Survey of their Random Processes. *SIAM Review* 27, 335–403.

[29] B.T. DOSHI AND H. HEFFES (1986). Overload Performance of Several Processor Queueing Disciplines for the M/M/1 Queue. *IEEE Trans. Comm.* COM-34, 538–546.

[30] T. DUNCAN AND W.H. HUEN (1982). Software Structure of No. 5 ESS — A Distributed Telephone Switching System. *IEEE Trans. Comm.* COM-30, 1379–1385.

[31] T. ENGSET (1918). The Probability Theory for Computing the Number of Switching Equipments in Automatic Telephone Exchange. *E.T.Z.* 31, 304–305. (In German).

[32] A. EPHREMIDES, P. VARAIYA, AND J. WALRAND (1980). A Simple Dynamic Routing Problem. *IEEE Trans. Autom. Control* AC-25, 690–693.

[33] A. EPHREMIDES AND S. VERDÚ (1989). Control and Optimization Methods in Communication Network Problems. *IEEE Trans. Autom. Control* AC-34, 930–942.

[34] A.K. ERLANG (1917). Solution of Some Problems in the Theory of Probabilities of Significance in Automatic Telephone Exchanges. *Post Office Electrical Engineer's Journal* 10, 189–197.

[35] G. FAYOLLE AND M.A. BRUN (1988). On a System with Impatience and Repeated Calls, in *Queueing Theory and its Applications, Liber Amicorum for J.W. Cohen*, 283–305, O.J. Boxma and R. Syski (eds.), North-Holland, Amsterdam.

[36] L.J. FORYS (1983). Performance analysis of a new overload strategy, in *Proc. of the 10th International Teletraffic Congress.*

[37] G.J. FOSCHINI AND B. GOPINATH (1983). Sharing Memory Optimally. *IEEE Trans. Comm.* COM-31, 352–359.

[38] G.H. GOLUB AND C.F. VAN LOAN (1983). *Matrix Computations.* The John Hopkins University Press, Baltimore.

[39] W.J. GORDON AND G.F. NEWELL (1967). Closed Queueing Systems with Exponential Servers. *Oper. Res.* 15, 254–265.

[40] B. HAJEK (1984). Optimal Control of Two Interacting Service Stations. *IEEE Trans. Autom. Control* AC-29, 491–499.

[41] D.P HEYMAN AND M.J. SOBEL (1984). *Stochastic Models in Operations Research, Volume II: Stochastic Optimization.* McGraw-Hill, New York.

[42] A. HORDIJK AND N.M. VAN DIJK (1981). Networks of Queues with Blocking, in *Performance '81*, 51–65, F.J. Kylstra (ed.), North-Holland, Amsterdam.

[43] A. HORDIJK AND N.M. VAN DIJK (1981). Stationary Probabilities for Networks of Queues, in *Applied Probability - Computer Science, The Interface, vol. II*, 423–451, R.L. Disney and T.J. Ott (eds.), Birkhauser, Boston.

[44] A. HORDIJK AND N.M. VAN DIJK (1983). Networks of Queues, in *Modelling and Performance Evaluation Methodology*, 151–205, F. Baccelli and G. Fayolle (eds.), Springer-Verlag, Berlin.

[45] A. HORDIJK AND F. SPIEKSMA (1989). Constrained Admission Control to a Queueing System. *Adv. Appl. Prob.* 21, 409–431.

[46] J.R. JACKSON (1957). Networks of Waiting Lines. *Oper. Res.* 5, 518–521.

[47] S.G. JOHANSEN AND S. STIDHAM, JR. (1980). Control of Arrivals to a Stochastic Input-Output System. *Adv. Appl. Prob.* 12, 972–999.

[48] B. KARLANDER (1973). Control of central processor load in an SPC system. *Ericsson Technics* 30, 221–243.

[49] J.S. KAUFMAN (1981). Blocking in a Shared Resource Environment. *IEEE Trans. Comm.* COM-29, 1474–1481.

[50] F.P. KELLY (1979). *Reversibility and Stochastic Networks.* Wiley, New York.

[51] D.G. KENDALL (1954). Stochastic Processes Occurring in the Theory of Queues and Their Analysis by the Method of the Embedded Markov Chain. *Ann. Math. Stat.* 34, 338–354.

[52] L. KLEINROCK (1967). Time-shared Systems: A Theoretical Treatment. *J. Assoc. Comput. Mach.* 14, 242–261.

[53] L. KLEINROCK (1975). *Queueing Systems, Vol. I: Theory.* Wiley, New York.

[54] L. KLEINROCK (1976). *Queueing Systems, Vol. II: Computer Applications.* Wiley, New York.

[55] G.P. KLIMOV (1974). Time-Sharing Service Systems, I. *Theory Probab. Appl.* 19, 532–551.

[56] A.E. KRZESINSKI (1987). Multiclass Queueing Networks with State-Dependent Routing. *Perf. Eval.* 7, 125–143.

[57] P.R. KUMAR AND P. VARAIYA (1986). *Stochastic Systems: Estimation, Identification and Adaptive Control.* Prentice-Hall, Englewood Cliffs, New Jersey.

[58] S.S. LAM (1977). Queueing Networks with Population Size Constraints. *IBM J. Res. Develop.* 21, 370–378.

[59] S.S. LAVENBERG (1989). A Perspective on Queueing Models of Computer Performance. *Perf. Eval.* 10, 53–76.

[60] W. LIN AND P.R. KUMAR (1984). Optimal Control of a Queueing System with Two Heterogeneous Servers. *IEEE Trans. Autom. Control* AC-29, 696–703.

[61] S.A. LIPPMAN (1975). Applying a New Device in the Optimization of Exponential Queueing Systems. *Oper. Res.* 23, 687–710.

[62] J.D. LITTLE (1961). A Proof of the Queueing Formula L=λW. *Oper. Res.* 22, 417–421.

[63] D.-J. MA AND A.M. MAKOWSKI (1987). Optimality Results for a Simple Flow Control Problem, in *Proc. of the 26th Conf. on Decision and Control*, 1852–1857, IEEE Press, Piscataway.

[64] P. NAIN (1989). Interchange Arguments for Classical Scheduling Problems in Queues. *Systems Control Lett.* 12, 177–184.

[65] P. NAIN AND K.W. ROSS (1986). Optimal Multiplexing of Heterogeneous Traffic with Hard Constraint. *Performance Evaluation Review* 14, 100–108.

[66] P. NAIN AND K.W. ROSS (1986). Optimal Priority Assignment with Hard Constraint. *IEEE Trans. Autom. Control* AC-31, 883–888.

[67] P. NAIN, P. TSOUCAS, AND J. WALRAND (1989). Interchange Arguments in Stochastic Scheduling. *J. Appl. Prob.* 26, 815–826.

[68] R.O. ONVURAL AND H.C. PERROS (1986). On Equivalencies of Blocking Mechanisms in Queueing Networks with Blocking. *Oper. Res. Letters* 5, 293–297.

[69] B. PITTEL (1979). Closed Exponential Networks of Queues with Saturation. The Jackson-type Stationary Distribution and its Asymptotic Analysis. *Math. Oper. Res.* 4, 357–378.

[70] M.L. PUTERMAN AND M.C. SHIN (1978). Modified Policy Iteration Algorithms for Discounted Markov Decision Problems. *Management Sci.* 24, 1127–1137.

[71] M. REISER (1979). Mean Value Analysis of Queueing Networks: A New Look at an Old Problem, in *Fourth Int. Symp. on Modelling and Performance Evaluation of Computer Systems*, 63–77.

[72] M. REISER AND H. KOBAYASHI (1975). Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms. *IBM J. Res. Develop.* 19, 283–294.

[73] M. REISER AND S.S.LAVENBERG (1980). Mean-Value Analysis of Closed Multichain Queueing Networks. *J. Assoc. Comput. Mach.* 27, 313–322.

[74] T.G. ROBERTAZZI AND A.A. LAZAR (1985). On the Modeling and Optimal Flow Control of the Jacksonian Network. *Perf. Eval.* 5, 29–43.

[75] K.W. ROSS (1985). *Constrained Markov Decision Processes with Queueing Applications.* PhD thesis, Computer, Information and Control Engineering, University of Michigan.

[76] S.M. ROSS (1970). *Applied Probability Models with Optimization Applications.* Holden-Day, San Francisco.

[77] S.M. ROSS (1983). *Introduction to Stochastic Dynamic Programming.* Academic Press, New York.

[78] I. RUBIN AND J.K. LEE (1988). Performance Analysis of Interconnected Metropolitan Area Circuit-Switched Telecommunications Networks. *IEEE Trans. Comm.* COM-36, 171–185.

[79] R. SCHASSBERGER (1978). Insensitivity of Steady State Distributions of Generalised Semi-Markov Processes with Speeds. *Adv. Appl. Prob.* 10, 836–851.

[80] F.C. SCHOUTE (1981). Optimal Control and Call Acceptance in a SPC Exchange, in *Proc. of the 9th International Teletraffic Congres.*

[81] F.C. SCHOUTE (1983). The Hierarchical Queue: A Model for Definition and Estimation for Processor Loading. Report SR2200-83-3743, Philips Telecommunication and Data Systems, Hilversum.

[82] F.C. SCHOUTE (1983). Overload Control in SPC Processors. *Philips Telecommunication Review* 41, 300–310.

[83] F.C. SCHOUTE (1984). Overload Control in Telephony Processors, in *Proc. NGI-SION Symposium 2*, 509–516.

[84] M. SCHWARTZ (1987). *Telecommunication Networks: Protocols, Modeling and Analysis.* Addison-Wesley, Reading.

[85] L.I. SENNOTT (1986). A New Condition for the Existence of Optimal Stationary Policies in Average Cost Markov Decision Processes. *Oper. Res. Letters* 5, 17–23.

[86] L.I. SENNOTT (1989). Average Cost Optimal Stationary Policies in Finite State Markov Decision Processes with Unbounded Costs. *Oper. Res.* 37, 626–633.

[87] L.I. SENNOTT (1989). Average Cost Semi-Markov Decision Processes and the Control of Queueing Systems. *Probab. Engin. Inform. Sci.* 3, 247–272.

[88] R.F. SERFOZO (1979). An Equivalence between Continuous and Discrete Time Markov Decision Processes. *Oper. Res.* 27, 616–620.

[89] R.F. SERFOZO (1988). Markovian Network Processes with System-Dependent Transition Rates. Research report, Georgia Institute of Technology, Atlanta.

[90] S. STIDHAM, JR. (1978). Socially and Individually Optimal Control of Arrivals to a GI/M/1 Queue. *Management Sci.* 24, 1598–1610.

[91] S. STIDHAM, JR. (1985). Optimal Control of Admission to a Queueing System. *IEEE Trans. Autom. Control* AC-30, 705–713.

[92] S. STIDHAM, JR. AND N.U. PRABHU (1974). Optimal Control of Queueing Systems, in *Mathematical Methods in Queueing Theory*, 263–294, A.B. Clarke (ed.), Springer-Verlag, Berlin.

[93] D. STOYAN (1983). *Comparison Methods for Queues and Other Stochastic Models.* Wiley, New York.

[94] C. STRIEBEL (1975). *Optimal Control of Discrete Time Stochastic Systems*, vol. 110 of *Lecture Notes in Economics and Mathematical Systems.* Springer-Verlag, Berlin.

[95] R. SURI (1989). Perturbation Analysis: The State of the Art and Research Issues Explained via the G/G/1 Queue. *Proc. IEEE* 77, 114–137.

[96] D.Y. SZE (1986). A Queueing Model for Overload Analysis, in *IFIP WG 7.3 International Seminar on Computer Networking and Performance Evaluation*, 413–422, T. Hasegawa, H. Takagi, and Y. Takahashi (eds.), North-Holland, Amsterdam.

[97] H.C. TIJMS (1986). *Stochastic Modelling and Analysis: A Computational Approach.* Wiley, Chichester.

[98] D. F. TOWSLEY (1980). Queueing Network Models with State-Dependent Routing. *J. Assoc. Comput. Mach.* 27, 323–337.

[99] P. TRAN-GIA AND M.H. VAN HOORN (1986). Dependency of Service Time on Waiting Time in Switching Systems - A Queueing Analysis with Aspects of Overload Control. *IEEE Trans. Comm.* COM-34, 357–364.

[100] P.R. DE WAAL (1987). Performance Analysis and Optimal Control of an M/M/1/k Queueing System with Impatient Customers, in *Messung, Modellierung und Bewertung von Rechensystemen, 4. GI/ITG-Fachtagung, Erlangen*, 28–40, U. Herzog and M. Paterok (eds.), Springer-Verlag, Berlin.

[101] P.R. DE WAAL AND N.M. VAN DIJK (1989). Interconnected Networks of Queues with Randomized Arrival and Departure Blocking. Report BS-R8934, Centre for Mathematics and Computer Science, Amsterdam. (submitted to *J. Assoc. Comput. Mach.*).

[102] P.R. DE WAAL AND N.M. VAN DIJK (1989). Monotonicity of Performance Measures in a Processor Sharing Queue. Report BS-R8902, Centre for Mathematics and Computer Science, Amsterdam. (to appear in *Perf. Eval.*).

[103] P.R. DE WAAL, P. NAIN, D.F. TOWSLEY, AND J.P.C. BLANC (1990). A New Device for the Synthesis Problem of Optimal Admission Control in an M/M/c Queue. tentative title.

[104] J. WALRAND (1984). A Note on Optimal Control of a Queuing System with Two Heterogeneous Servers. *Systems Control Lett.* 4, 131–134.

[105] J. WALRAND (1988). *An Introduction to Queueing Networks.* Prentice-Hall, Englewood Cliffs, New Jersey.

[106] P. WHITTLE (1985). Partial Balance and Insensitivity. *J. Appl. Prob.* 22, 168–175.

[107] P. WHITTLE (1986). *Systems in Stochastic Equilibrium.* Wiley, New York.

[108] D.V. WIDDER (1941). *The Laplace Transform.* Princeton University Press, Princeton.

[109] D.D. YAO AND J.A. BUZACOTT (1987). Modeling of a Class of Flexible Manufacturing Systems with Reversible Routing. *Oper. Res.* 35, 87–93.

Samenvatting

In dit proefschrift wordt de analyse en synthese van regelwetten beschreven voor computergestuurde (SPC) telefooncentrales in overbelasting. In een SPC centrale zorgt een processor, gestuurd door een speciaal ontwikkeld computerprogramma, voor het tot stand brengen van verbindingen. Voor een gespreksaanvraag moet de processor een groot aantal deeltaken uitvoeren, zoals bijvoorbeeld het genereren van een kiestoon of belsignaal en de analyse van het gekozen nummer. Deze taken samen worden genoemd de opbouw van een verbinding. De verwerking van deze deeltaken van de opbouw geschiedt volgens "time-sharing", wat wil zeggen dat de processor een aantal gespreksaanvragen tegelijk kan behandelen. Aangezien de processor slechts een eindige capaciteit heeft, betekent dit dat de opbouwtijd per verbinding toeneemt als er meer aanvragen tegelijkertijd behandeld moeten worden.

Een belangrijke grootheid voor het prestatievermogen van een centrale is de effectieve capaciteit, dat wil zeggen het aantal verbindingen dat per tijdseenheid tot stand gebracht wordt. Wanneer de vraag naar verbindingen groter is dan de centrale kan verwerken, spreken we van overbelasting. Uit waarnemingen is gebleken dat, indien geen regeling wordt toegepast, de effectieve capaciteit van een centrale in overbelasting sterk vermindert. Deze teruggang van het prestatievermogen van een centrale wordt voornamelijk veroorzaakt door het vroegtijdig afbreken van gespreksaanvragen vóór een verbinding tot stand gebracht is. De belangrijkste oorzaak voor dit afbreken is een te lange opbouwtijd. Wanneer de opbouwtijd groot wordt, zullen een aantal abonnees een aanvraag in opbouw afbreken, omdat hun geduld opraakt voordat het gebelde toestel hoorbaar overgaat. Het werk dat de processor aan een afgebroken aanvraag besteed heeft, mag als nutteloos beschouwd worden, aangezien dit niet tot een succesvolle verbinding leidt. Dit gaat ten koste van de effectieve capaciteit. Doordat ongeduldige abonnees mogelijk een herhaalde oproep plegen, gaat de aankomstintensiteit nog verder omhoog. In het uiterste geval kan het voorkomen dat alle abonnees vroegtijdig afhaken, waarbij de processor op volle snelheid werkt, doch geen enkele verbinding tot stand gebracht wordt. Het mag duidelijk zijn dat in het belang van zowel de gebruikers als de exploitant van de centrale het ineenstorten van de effectieve capaciteit voorkomen moet worden.

In dit proefschrift wordt de aanzet gegeven tot de ontwikkeling en analyse van regelwetten voor telefooncentrales in overbelasting. Hierbij wordt gebruik gemaakt van methoden uit de wachtrijtheorie en de stochastische regeltheorie. De bestudering van de regeling van een centrale kan op twee manieren worden aangepakt. Enerzijds kan men prestatiematen berekenen van een wachtrijmodel met een gegeven regeling. Voor deze prestatiematen kan men criteria formuleren waaraan de centrale moet voldoen om goed te functioneren. Met behulp van prestatie-analyse van een wachtrijmodel

met regeling kan men vervolgens proberen een regeling te vinden die aan deze criteria voldoet. Anderzijds kan men met behulp van Markov beslissingstheorie proberen de optimale regeling te vinden van een wachtrijmodel met een gegeven opbrengststructuur. In dit geval wordt het prestatievermogen van een centrale geformuleerd als een verdisconteerde of tijdsgemiddelde opbrengst en probeert men een regeling te vinden die het prestatievermogen optimaliseert.

In Hoofdstuk 1 wordt een korte inleiding en motivatie geschetst. Hierin ligt de nadruk op het belang van het correcte functioneren van communicatie-netwerken in de hedendaagse samenleving.

In Hoofdstuk 2 beschrijven we de werking van een computergestuurde centrale. Hierin wordt uitgelegd welke taken een centrale moet verrichten om abonnees in staat te stellen andere abonnees te bellen. Bovendien schetsen we in dit hoofdstuk wat er gebeurt als de centrale overbelast wordt. Het hoofdstuk besluit met een paragraaf waarin aandacht besteed wordt aan verschillende aspekten van regeling van een centrale in overbelasting.

Hoofdstuk 3 bevat het wiskundige kader voor de overige hoofdstukken. Als wiskundig model voor een centrale kiezen we een netwerk van wachtrijen. In Hoofdstuk 3, dat uit twee gedeelten bestaat, wordt eerst aandacht besteed aan prestatie-analyse van wachtrijnetwerken met regeling van aankomst- en vertrekprocessen. Aangetoond wordt onder welke voorwaarden de evenwichtsverdeling van zo'n netwerk een produktvorm heeft. Het bestaan van zo'n produktvorm kan de prestatie-analyse van een wachtrijnetwerk sterk vereenvoudigen. In het tweede gedeelte van Hoofdstuk 3 worden Markov beslissingsproblemen voor wachtrijsystemen behandeld. Hier wordt ondermeer aandacht besteed aan het afleiden van structurele eigenschappen van optimale regelwetten, daar deze eigenschappen nuttig zijn bij het bepalen van de optimale regeling.

In Hoofdstuk 4 worden de resultaten uit het voorafgaande hoofdstuk over optimale regeling toegepast in een model van een wachtrij met ongeduldige klanten. Voor dit model wordt een Markov beslissingsprobleem geformuleerd waarin we proberen een toelatingsregeling te ontwerpen die de effectieve capaciteit van de centrale maximaliseert. We tonen aan dat onder een aantal zwakke voorwaarden de optimale regeling een drempelregeling is. Zo'n regeling beperkt het aantal gespreksaanvragen dat tegelijkertijd in behandeling kan zijn tot een drempelwaarde. De optimaliteit van deze klasse van regelingen wordt aangetoond met behulp van een nieuwe methode, waarbij structurele eigenschappen van optimale regelwetten direkt uit de Dynamisch Programmeringsvergelijkingen worden afgeleid.

In Hoofdstuk 5 worden de twee wiskundige methoden uit Hoofdstuk 3 gecombineerd voor de bestudering van een gedetailleerd model van een centrale. In dit model worden zowel gespreksaanvragen als operator taken aan de centrale voor verwerking aangeboden. Operator taken zijn taken die niet direkt verband houden met het tot stand brengen van verbindingen, maar wel noodzakelijk zijn voor het correcte functioneren van de centrale. Er wordt een prestatie-analyse van twee klassen van toelatingsregelingen gegeven en voor een van deze twee klassen wordt een regelprobleem met nevenvoorwaarden opgelost.

Een hoofdstuk met conclusies en suggesties voor toekomstig onderzoek besluit dit proefschrift.

List of Figures

List of Tables

Index